T. C.
BAHÇEŞEHİR UNIVERSITY

WAREHOUSE MODELLING AND VERIFICATION FOR DECISION SUPPORT SYSTEM: TRANSPORTATION SYSTEM

M. S. Thesis

Mert SUN

İstanbul, 2011

T. C.
BAHÇEŞEHİR UNIVERSITY
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program

WAREHOUSE MODELLING AND VERIFICATION FOR DECISION SUPPORT SYSTEM: TRANSPORTATION SYSTEM

M. S. Thesis

Mert SUN

Supervisor: Prof. Oya KALIPSIZ

İstanbul, 2011

T.C
BAHÇEŞEHİR ÜNİVERSİTESİ
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program

Title of the Master's Thesis	: Warehouse Modelling and Verification for Decision Support System: Transportation System
Name/Last Name of the Student	: Mert SUN
Date of Thesis Defense	: 23.08.2011

The thesis has been approved by the Graduate School of Natural and Applied Sciences

Assoc. Prof. F. Tunç BOZBURA

Acting Director

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Committee Members:

Prof. Oya KALIPSIZ (Supervisor) :

Assist. Prof. Övgü Öztürk :

Assist. Prof. Egemen Özden :

ABSTRACT

WAREHOUSE MODELLING AND VERIFICATION FOR DECISION SUPPORT SYSTEM: TRANSPORTATION SYSTEM

Sun, Mert

Computer Engineering Graduate Program

Supervisor: Prof. Oya KALIPSIZ

August, 2011, 109 pages

In the competitive transportation sector, there's a great need for managers to analyze the data of business processes without interrupting the daily work of an On-Line Transaction Processing (OLTP) system. The new technologies and concepts have given the opportunity to the companies for analyzing faster and more detailed results with using the Decision Support System (DSS). Improvements in the data warehousing technologies expanded the view of the business about the decision making. With the On-Line Analysis Processing (OLAP) concepts of the data warehouse, presentation of the data through multidimensional views and graphical displays provide great support for the decision-makers. With new DSS technologies, transportation companies will get chance to analyze their business more efficiently. In this thesis, we analyzed the structure of OLTP of a transportation company and we proposed the most suitable DSS model as a data warehouse based model. For the DSS data warehouse database, we modeled a hybrid structure which is suitable for the business functions' data of the transportation company.

Keywords: Warehouse systems, transportation company systems, data warehouse hybrid structure

ÖZET

KARAR DESTEK SİSTEMLERİ İÇİN VERİAMBARI MODELLEMESİ VE GERÇEKLEMESİ: TAŞIMACILIK SİSTEMİ

Sun, Mert

Bilgisayar Mühendisliği Master Programı

Danışman: Prof. Oya KALIPSIZ

Ağustos, 2011, 110 sayfa

Rekabete dayalı taşımacılık sektöründe, yöneticilerin günlük işleyişte kullanılan çevirmiçi transaksiyonel işlem sistemlerini (OLTP) etkilemeden, iş süreçlerine ait verinin analizini yapmaya çok önemli derecede ihtiyaçları vardır. Yeni teknolojiler ve konseptler, Karar Destek Sistemleri'ni (DSS) kullanacak olan şirketlere daha hızlı ve daha detaylı analiz sonuçlarına ulaşabilmek için fırsatlar sunmaktadır. Veri ambarı teknolojilerindeki gelişmeler iş dünyasının karar destek kavramlarıyla ilgili bakış açısını genişletmiştir. Online analiz proses (OLAP) konseptleri sayesinde veriambarlarının sağladığı, verinin çok boyutlu ve grafiksel gösterim imkanı, karar destek uzmanlarına çok önemli destek sağlamıştır. Yeni DSS teknolojilerinden faydalanacak olan taşımacılık sektöründeki şirketler, kendi iş süreçlerinin analizini çok daha etkili yapabilme şansına sahip olabileceklerdir. Bu tezde, bir taşımacılık şirketinin OLTP yapısı analiz edildi ve bu yapıya en uygun olacak, veritabanı bazlı bir DSS modeli geliştirildi. Taşımacılık şirketinin iş süreçleri için kullandığı veri göz önünde bulundurularak, hibrit yapıda bir veri ambarı DSS sistemi modellendi.

Anahtar Kelimeler: Veri ambarı sistemi, taşımacılık şirketi sistemleri, hibrit modelli veri ambarı yapısı

CONTENTSS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The business in transportation sector is constantly changing and developing. Day after day it's becoming more complex. Organizations are under pressures to respond quickly to these changing conditions. With the growing of the sector, new products are developed such as new transportation methods. Companies have to be agile and they must make quick strategic, tactical, and operational decisions which are very complex. For making such decisions, considerable amounts of relevant data, information and knowledge is required. When preparing these data, processing must be done quickly, frequently and usually requires some computerized support.

Competition in the transportation sector today is based not just on price but also on quality, timeliness, customization of products, and customer support. In addition, organizations must be able to frequently and rapidly change their mode of operation, reengineer processes and structures, and innovate in order to adapt to their changing environments. Decision support technologies such as intelligent systems can empower people by allowing them to make good decisions quickly, even if they lack some knowledge. (Turban, Sharda, Delen 2007)

The new technologies and concepts are always developint in the IT world. Computer applications have moved from transational processing and monitoring activities to problem analysis and solution applications, and much of the activity is done with Web-based technologies. IT tools such as data warehousing, Online Analytical Processing (OLAP), dashboards, and the use of the Web for decision support are the cornerstones of today's modern management. Managers must have high-speed, networked information systems such as wireless systems to assist them with their most important task: making decisions.

Since the Internet and Web servers and tool's development, there have been dramatic changes in how decision makers are supported. Most important, the Web provides access to a vast body of data, information, and knowledge available around the world. It

also provides a common, user-friendly graphical user interface that is easy to learn to use and readily available.

DSS enables the managers to perform many analysis quickly and at a low cost. Web-based DSS can improve the collaboration process of a group and enable its members who are in different places. In addition of that, DSS can increase the productivity of staff support. With DSS, extremely great data of the business can be analyzed. (Turban, Sharda, Delen 2007) With using wireless technology, managers can access information anytime and from anyplace, analyze and interpret it.

In this thesis, a transportation company is analyzed. The company's needs are determined. The problems which the company has are listed. With these analysis, the decision subjects are defined.

For covering decision makers' needs of the transportation sector, the modern data warehouse based DSS are analyzed. Also, the best structure of the data warehouse is researched which is most suitable to the transportation sector's business needs and the company's OLTP systems.

With choosing the best suitable techniques, the data warehouse structure is modeled for the company. For developing analyzing layer, the analyzing tools are compared and the most suitable tool is selected. With the analyzing tool, business intelligence layer is structured.

This thesis consists of five chapter. In the first chapter, DSS are defined and their benefits are analyzed. Their differences between the OLTP systems are researched. The framework and the structure of the DSS is explained.

In the second chapter, data warehouse is researched. The features, structures and architectures of the data warehouse are explained. Also the ETL processes are analyzed. Data warehouse's database design and models are researched.

In the third chapter, transportation company's DSS need is analyzed. The problems about making decisions are researched.

In the forth chapter, the transportation company is analyzed. It's functional needs are defined. The problems that the transportation company has are defined.

In the fifth chapter, according to the analysis of the company's needs and the OLTP system's structure, most suitable structure of the data warehouse is researched. ETL processes are developed. Data warehouse analyzing objects are designed. OLAP Cube's of the DSS are developed. Reports and dashboards are explained. Data warehouse DSS is verified. The modeled DSS system and OLTP systems are compared. The benefits of the data warehouse DSS model is defined.

2. DECISION SUPPORT SYSTEM (DSS)

2.1 DECISION SUPPORT SYSTEM (DSS)

A Decision Support System (DSS) is an interactive computer based system which helps decision makers utilize data and models to solve unstructured problems. (Keen and Scott-Morton (1978))

Decision Support Systems include knowledge-based systems. A DSS is an interactive software-based system intended to help decision makers compile useful information from a combination of raw data, documents, personal knowledge, or business models to identify and solve problems and make decisions. (Power, 2004).

An important role of a Decision Support System is to provide information for users to analyze situations and make decisions. A Decision Support System provides information for employees to make decisions and do their jobs more effectively. (V. Poe, 1998)

Decision Support Systems are used to collect data, analyze and shape the data that is collected, and make sound decisions or construct strategies from analysis. Whether computers, databases, or people are involved usually does not matter.
Typical information that a decision support application might gather and present would be:

- Accessing all of your current information assets, including legacy and relational data sources, cubes, data warehouses
- Comparative sales figures between one week and the next
- Projected revenue figures based on new product sales assumptions
- The consequences of different decision alternatives, given past experience in a context that is described

2.2 NEEDS FOR DSS

Every one of the past attempts at providing strategic information to decision makers was unsatisfactory. The cycle of strategic information provision in the past always revolves in these phases:

a. User needs information
b. User requests reports from IT
c. IT places request on backlog
d. IT creates ad hoc queries
e. IT sends requested reports

Here are some of the factors relating to the inability to provide strategic information:

- IT receives too many ad hoc requests, resulting in a large overload. With limited resources, IT is unable to respond to the numerous requests in a timely fashion.
- Requests are not only too numerous, they also keep changing all the time. The users need more reports to expand and understand the earlier reports.
- The users find that they get into the spiral of asking for more and more supplementary reports, so they sometimes adapt by asking for every possible combination, which only increases the IT load even further.
- The users have to depend on IT to provide the information. They are not able to access the information themselves interactively.
- The information environment ideally suited for making strategic decision making has to be very flexible and conducive for analysis. IT has been unable to provide such an environment. (Ponniah, 2001)

2.3 DIFFERENCE BETWEEN DSS AND OLTP

Online transaction processing (OLTP) database applications are optimal for managing changing data. OLTP uses operational data. These applications typically have many users who are performing transactions at the same time that change real-time data, in other words OLTP is a live database (accomodates inserts, deletes, updates etc). (MSDN 2005). But DSS is used for making analysis of the organization's data. Because

of the differences in usage of these two systems, their structures and characteristics are also different. The basic differences between DSS and OLTP are shown in Table 2.1.

PRIMITIVE DATA/OPERATIONAL DATA	DERIVED DATA/DSS DATA
• Application-oriented	• Subject-oriented
• Detailed	• Summarized, otherwise refined
• Accurate, as of the moment of access	• Represents values over time, snapshots
• Serves the clerical community	• Serves the managerial community
• Can be updated	• Is not updated
• Run repetitively	• Run heuristically
• Requirements for processing understood *a priori*	• Requirements for processing not understood *a priori*
• Compatible with the SDLC	• Completely different life cycle
• Performance-sensitive	• Performance relaxed
• Accessed a unit at a time	• Accessed a set at a time
• Transaction-driven	• Analysis-driven
• Control of update a major concern in terms of ownership	• Control of update no issue
• High availability	• Relaxed availability
• Managed in its entirety	• Managed by subsets
• Nonredundancy	• Redundancy is a fact of life
• Static structure; variable contents	• Flexible structure
• Small amount of data used in a process	• Large amount of data used in a process
• Supports day-to-day operations	• Supports managerial needs
• High probability of access	• Low, modest probability of access

Table 2.1 : Differences between operational and derived data (Inmon, 2005)

2.4 HISTORICAL DEVELOPMENT OF DSS

Since the beginning of the organization of business processes into functions that optimized record keeping and thereby the ability to compete successfully in the marketplace, there has been a need to display or report on the basic information that was used by the direct functional processes. The business processes were distilled and encoded in programming languages that provided a concise set of actions to be performed on the data. The data used for the processes were arranged in the most optimal structure possible to ensure rapid movement through the programs that represented the captured business processes. (Tupper, 2009).

The structure of the data for optimal processing for the business did not represent some of the information (interpreted data) necessary to monitor or project trends in the business. The structure that allowed rapid processing of transaction-type activity

impeded the process of interpreting the information and arranging it in a format that allowed business decisions to be based on it.

The early DBMSs (database management systems) did not help the situation, since they tended to be inflexible and required that the data be arrayed in a pattern that the processing requirements for a specific business process needed. If other business processes needed that same data, then their business need was captured in a separate data structure.

Peter G. W. Keen and Michael S. Scott-Morton (1978) developed some concepts of business decision classification and decision support strategies for use in reporting and projective analysis. Theirs was the first comprehensive look at the business need to provide intelligence on the processing of the data for monitoring and control purposes. In their work on decision support they identified three classes of decisions: structured decisions, semistructured decisions, and unstructured decisions. (Tupper, 2009).

Structured decisions were made by operating management. Because they were regarded as needing certain expertise to be accomplished. We know now that these decisions are easily automated and generally choose to computerize them.

Semistructured decisions are less easily automated because they rely on judgment, intuition, and experience of management. The data that are needed for these semistructured decisions usually lies in the detail data of the business processes and can be retrieved for interpretation.

Unstructured decisions are decisions that rely completely on human intuition and analysis. The data needed for these must be formulated and structured for the purpose of presentation for evaluation, analysis, and assessment.

The structured decision classification was the set of data currently used for transaction processing systems. The set of data that is applicable for semistructured decisions is what is considered as reporting system data. And finally, the set of data associated with the classification of unstructured decisions is regarded as ad hoc query data.

Ralph Sprague and Eric Carlson's(1982) book Building Effective Decision Support Systems was an important milestone that provided a practical overview of how organiations could and should build DSS. (Power, 2002).

In the early 1990's, a shift occured from mainframe-based data-driven DSS to client/server DSS. Some desktop OLAP tools were introduced at this period. In 1992-1993, vendors recommended object-oriented technology for building "re-usable" decision support capabilities. Also, some of the first data warehouse were completed.

In 1994, many companies started to upgrade their network infrastructures. Database Management System vendors changed their focus from On-Line Transaction Processing (OLTP) and recognized that decision support was different from OLTP and started implementing real OLAP capabilities into their databases (Powel, 2004)

The modern era in decision support systems started in about 1995 with the specification of HTML 2.0, the expansion of the World Wide Web in companies, and the introduction of handheld computing. (Power, 2009).

In 1997, the data warehouse became the cornerstone of an intergrated knowledge environment that provided a higher level of information sharing across an organization, enabling faster and better decision making. In 1998, enterprise poerformence management and balanced scorecard systems were introduced to update the executive information systems of the 1970s and 1980s.

In 2000 and 2001, application service providers (ASPs) began hosting some application software and some of the techical infrastructure for decision support capabilities. More sophisticated decision portals have also been introduced that combine information portals, knowledge management, business intelligence, and communications-driven DSS in an integrated Web enviroment.

Today, the Web 2.0 technologies, mobile-integrated communication and computing devices, and improved software development tools have revolutionized DSS user interfaces. Additionally, the decision support data store back-end is now capable of rapidly processing very large data sets. (Power, 2009).

Modern DSS are more complex and more diverse in functionality than DSS built prior to the widespread use of the World Wide Web. Today, we are seeing more decision automation with business rules and more knowledge-driven decision support systems. Current DSS are changing the mix of decision-making skills needed in organizations. Building better DSS may provide one of the "keys" to competing in a global business environment.

The following attributes are increasingly common in new and updated decision support systems. Some attributes are more closely associated with one category of DSS, but sophisticated DSS often have multiple subsystems. Attributes of contemporary DSS include the following:

a. Multiple, remote users can collaborate in real-time using rich media.
b. Users can access DSS applications anywhere and anytime.
c. Users have fast access to historical data stored in very large data sets.
d. Users can view data and results visually with excellent graphs and charts.
e. Users can receive real-time data when needed.

2.5 DEVELOPMENT FRAMEWORKS OF DSS

DSS systems are not entirely different from other systems and require a structured approach. Such a framework includes people, technology, and the development approach.

DSS technology levels (of hardware and software) may include:

a. The actual application that will be used by the user. This is the part of the application that allows the decision maker to make decisions in a particular problem area. The user can act upon that particular problem.
b. Generator contains Hardware/software environment that allows people to easily develop specific DSS applications. This level makes use of case tools or systems such as Crystal, AIMMS, and iThink.
c. Tools include lower level hardware/software. DSS generators including special languages, function libraries and linking modules

An iterative developmental approach allows for the DSS to be changed and redesigned at various intervals. Once the system is designed, it will need to be tested and revised for the desired outcome. (Sprague, Carlson,1982)

2.6 BENEFITS OF DSS

DSS is the abbreviated form of Decision support systems and comprises of information systems based on a network of computers. DSS also includes knowledge-based systems, which support the decision-making activities in an organization. DSS supports the management of an organization and helps them in decision making. These decisions might be changing rapidly and are not specified in advance. There are many benefits of DSS both for the management and the organization as a whole. These benefits include:

a. Saves time: Research has demonstrated that decision support systems help to reduce decision cycle time for an organization. DSS provides timely information, which is then used for decision making and results in enhanced employee productivity.

b. Improves efficiency: Another advantage of DSS is efficient decision making, resulting in better decisions. This is because use of DSS results in quick transfer of information, better data analyses, thus resulting in efficient decisions.

c. Boosts up interpersonal communication: Use of DSS in an organization helps to improve interpersonal communication between same level of employees and between management and employees.

d. Provides competitive advantage: Use of decision support system in an organization provides a competitive advantage over other organizations which do not use DSS.

e. Helps in reducing cost: Research and case studies reveal that use of DSS in an organization helps in making quicker decisions and reduce cost.

f. High satisfaction among decision makers: In DSS computers and latest technology aids the decision making process. It thus results in higher satisfaction among decision makers, reduces frustrations among them, and form perceptions that superior information is being used. They gain a confidence and satisfaction that they are good decision makers.

g. Supports learning: The use of DSS in an organization results in two type of learning. First managers themselves learn new concepts. Secondly, there is better factual understanding of business as well as the decision making environment.

h. Enhanced organizational control: Due to the use of DSS business transaction data is easily available for monitoring the performance of employees and ad hoc querying. It thus leads to enhanced understanding of business operations for the management.

Although DSS has numerous advantages for organizations and the people involved in decision making, but should be used cautiously due to some associated disadvantages. As for instance some DSS development efforts can lead to power struggles. People fight over the authority of accessing data, thus spoiling the organizational environment. Sometimes managers may have some personal motives and may advocate the development of a particular DSS. This might harm other people and the organization as a whole. It should thus be very well and cautiously used in benefit of an organization.

2.7 DSS STRUCTURE

A typical DSS consists of three major parts: staging area, data warehouse and analytical part as shown in Figure 2.2

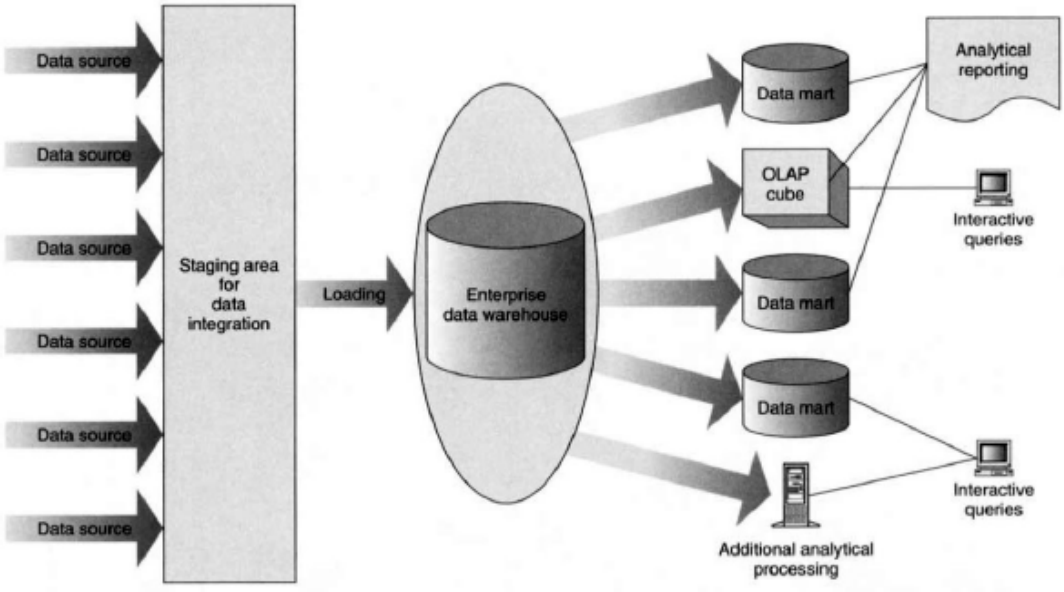

Figure 2.2 : DSS Structure (Loshin, 2003)

Staging area: Staging area is the first interface of the DSS. It serves as the collector and deliverer of the data to the data warehouse. It provides the interface to communicate with all the data sources and facilitates efficient and reliable ETL (Extraction, Transformation, Load) processing

Data Warehouse: The data warehouse is "the heart" of the whole system. Its role is to provide secure, long-term and accessible storage for granular data needed for analytical purposes. There is a strong dispute between the authors about the form in which the data should be stored in the data warehouse. One school of thought (e.g. Kimball) recommends storing the data directly in the multidimensional form within the data warehouse. The second school of thought (e.g. Inmon) believes that this approach is very shortsighted and does not provide a long-term, universal and efficient solution for an enterprise. They claim the data is mutilated and thus not versatile. They advise to use traditional relational approach and then add the third purpose-oriented analytical level to which the data is supplied from the data warehouse. In this paper, the latter solution will be preferred.

Analytical part: The third level is the end-user interface and it is the place where most of the DSS calculations and aggregations are performed. Typical unit is a datamart.

Datamart: Datamart serves the needs of departments or it is build around certain subject, such as production, sales, etc. The data has a higher level of summarization and it is aggregated based on its purpose. The number of datamarts depends on the company's needs and its size. The reason to have more than one datamart are different needs of different parts of the company. Every datamart gathers and aggregates information by different criteria and it would not be possible to facilitate all of this using only one datamart or the data warehouse solely. Although it may seem that there is unnecessary redundancy, it is a trade-off for a single consistent data foundation. The benefit is that the ETL is done only once and that all the analytical units work with the same data.

OLAP cube: On-Line Analytical Processing uses the data arranged in multidimensional structures to speed up the calculations and perform aggregations faster.

Operational Data Store (ODS): ODS is similar to a datamart with a slight difference, that the purpose of ODS is to gather profile information and serve as a high speed access unit to provide this information in an OLTP manner.

3. WAREHOUSE MODEL FOR THE DECISION SUPPORT SYSTEM

3.1 DATA WAREHOUSE

Data Warehouse (DW) is a pool of data that is produced to support decision-making is also a repository of current and historical data of potential interest to managers of the organization. The data are usually structured to be available in a form ready for the activities of analytical processing (ie, online analytical processing [OLAP], data mining, querying, reporting and other support applications the decision). A data warehouse is a subject oriented, integrated, time-varying, the non-volatile data collection to support management decision making. (Inmon, 2005)

A data warehouse is a collection of subject-oriented, integrated, non-volatile, and time-variant data to support management's decisions.

Data warehouses are not optimized for transaction processing, which is the domain of OLTP systems. Data warehouses usually consolidate historical and analytic data derived from multiple sources. Data warehouses separate analysis workload from transaction workload and enable an organization to consolidate data from several sources.

A data warehouse usually stores many months or years of data to support historical analysis. The data in a data warehouse is typically loaded through an extraction, transformation, and loading (ETL) process from one or more data sources such as OLTP applications, mainframe applications, or external data providers. (Oracle, 2007)

Organizations, private and public, continuously collect data, information, and knowledge at an increasingly accelerated rate and store them in computerized systems. Maintaining and using these data and information become extremely complex, especially as scalability issues arise. In addition, the number of users needing to access the information continues to increase as a result of improved reliability and availability of network access, especially the Internet. Working with multiple databases, either

integrated in a data warehouse or not, has become an extremely difficult task requiring considerable expertise, but it can provide immense benefits far exceeding its cost. (Inmon, 2005)

Data are imported from various external and internal resources anda re cleansed and organized in a manner consistent with the organization's needs. After the data are populated in the data warehouse, data marts can be loaded for a specific area or department. Alternatively, data marts can be created first, as needed, and then integrated into an EDW. Often, data marts are not developed, but data are simply loaded onto PC sor left in their original state for direct manipulation using BI tools. The data warehouse components is seen in Figure 3.1.

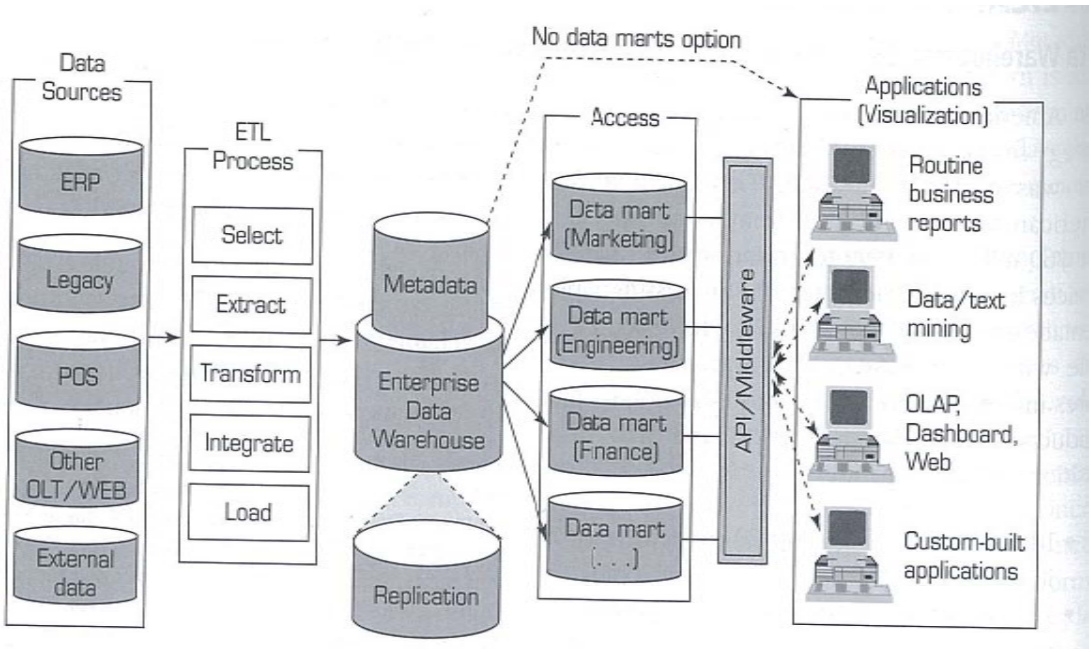

Figure 3.1 : Data warehouse components (Turban, 2007)

These are the major components of the data warehousing process:

Data Sources: Data are sourced from multiple independent operational systems and possibly from external data providers. Data may also come from OLTP (OnLine Transaction Processing) or ERP system. Web data in the form of Web logs may also feed a data warehouse.

Data extraction: Data are extracted using custom-written or commercial software called ETL.

Data loading: Data are loaded into a staging area, where they are transformed and cleansed. The data are then ready to load into the data warehouse.

Comprehensive database: Essentially, this is the EDW to support all decision analysis by providing relevant summarized and detailed information originating from many different sources.

Metadata: Metadata are maintained so that they can be assessed by IT poersonnel and users. Metadata include software programs about data and rules for organizing data summaries that are easy to index and search, especially with Web tools.

Middleware tools: Middleware tools enable access to the data warehouse. Power users such as analysts may write their own SQL queries. Others may employ a managed query environment to access data. There are many front-end applications that business users can use to interact with data stored in the data repositories, including data mining, OLAP, reporting tools, and data visualization tools.

Data mart: Data mart is the access layer of the data warehouse (DW) environment that is used to get data out to the users. The DM is a subset of the DW, usually oriented to a specific business line or team.

3.2 FEATURES OF DATA WAREHOUSE

Subject-oriented: In operational database, data is stored by individual applications. In the data sets for an order processing application, the data is kept for that particular application. These data sets provide the data for all the functions for entering orders, checking stock, verifying customer's credit, and assigning the order for shipment. But

these data sets contain only the data that is needed for those functions relating to this particular application.

In striking contrast, in the data warehouse, data is stored by subjects, not by applications. Business subjects differ from enterprise to enterprise. These are the subjects critical for the enterprise. For a manufacturing company, sales, shipments, and inventory are critical business subjects. For a retail store, sales at the check-out counter are critical subject. (Ponniah, 2010).

Integrated: For proper decision making, it's needed to pull together all the relevant data from the various applications. The data in the data warehouse comes from several operational systems.Source data are in different databases, files, and data segments. These are disparate applications, so the operational platforms and operating systems could be different. The file layouts, character code representations, and field naming conventions all could be different.

Before the data from various disparate sources can be usefully stored in a data warehouse, the inconsistencies must be removed. The various data elements must be standardized and make sure of the meanings of data names in each source application. Before moving the data into the data warehouse, we have to go through a process of transformation, consolidation, and integration of the source data.

Time-variant: For an operational system, the stored data contains the current values. In an accounts receivable system, the balance is the current outstanding balance in the customer's account. In an order entry system, the status of an order is the current status of the order. In a consumer loans application, the balance amount owed by the customer is the current amount. There's also some past transactions in operational systems, but, essentially, operational systems reflect current information because these systems support day-to-day current operations. (Ponniah, 2010).

On the other hand, the data in the data warehouse is meant for analysis and decision making. If a user is looking at the buying pattern of a specific customer, the user needs

data not only about the current purchase, but on the past purchases as well. (Ponniah, 2010).

A data warehouse, because of the very nature of its purpose, has to contain historical data, not just current values. Data is stored as snapshots over past and current periods. Every data structure in the data warehouse contains the time element. There are historical snapshots of the operational data in the data warehouse. This aspect of the data warehouse is quite significant for both the design and the implementation phases.

Nonvolatile: Data extracted from the various operational systems and pertinent data obtained from outside sources are transformed, integrated, and stored in the data warehouse. The data in the data warehouse is not intended to run the day-to-day business. When you want to process the next order received from a customer, you do not look into the data warehouse to find the current stock status. The operational order entry application is meant for that purpose. In the data warehouse, you keep the extracted stock status data as snapshots over time. You do not update the data warehouse every time you process a single order.

Data from the operational systems are moved into the data warehouse at specific intervals. Depending on the requirements of the business, these data movements take place twice a day, once a day, once a week, or once in two weeks. In fact, in a typical data warehouse, data movements to different data sets may take place at different frequencies. The changes to the attributes of the products may be moved once a week. Any revisions to geographical setup may be moved once a month. The units of sales may be moved once a day. You plan and schedule the data movements or data loads based on the requirements of your users.

Data Granularity: In an operational system, data is usually kept at the lowest level of detail. In a point-of-sale system for a grocery store, the units of sale are captured and stored at the level of units of a product per transaction at the check-out counter. In an order entry system, the quantity ordered is captured and stored at the level of units of a product per order received from the customer. Whenever you need summary data, you

add up the individual transactions. If you are looking for units of a product ordered this month, you read all the orders entered for the entire month for that product and add up. You do not usually keep summary data in an operational system.

When a user queries the data warehouse for analysis, he or she usually starts by looking at summary data. The user may start with total sale units of a product in an entire region. Then the user may want to look at the breakdown by states in the region. The next step may be the examination of sale units by the next level of individual stores. Frequently, the analysis begins at a high level and moves down to lower levels of detail.

In a data warehouse, therefore, you find it efficient to keep data summarized at different levels. Depending on the query, you can then go to the particular level of detail and satisfy the query. Data granularity in a data warehouse refers to the level of detail. The lower the level of detail, the finer the data granularity. Of course, if you want to keep data in the lowest level of detail, you have to store a lot of data in the data warehouse. You will have to decide on the granularity levels based on the data types and the expected system performance for queries.

3.3 DATA WAREHOUSE ARCHITECTURES

There are several basic architectures for data warehousing. Two-tier and three-tier architectures are common, but sometimes there is simply one tier. Hoffer divided the data warehouse into three parts: (Hoffer 2007)

a. The data warehouse itself, which contains the data and associated software.

b. Data acquisition (back-end) software, which extracts data from legacy systems and external sources, consolidates and summarizes them, and loads them into the data warehouse.

c. Client (front-end) software, which allows users to access and analyze data from the warehouse (a DSS/BI/business analytics (BA) engine)

In a three-tier architecture, operational systems contain the data and the software for data acquisition in one tier (i.e., the server), the data warehouse is another tier, and the

third tier includes the DSS/BI/BA engine (i.e., the application server) and the client.(Figure 3.2). Data from the warehouse are processed twice and deposited in an additional multidimensional database, organized for easy multidimensional analysis and presentation, or replicated in data marts. The advantage of the three-tier architecture is its separation of the funcitons of the data warehouse, which eliminates resource constraints and makes it possible to easily create data marts.

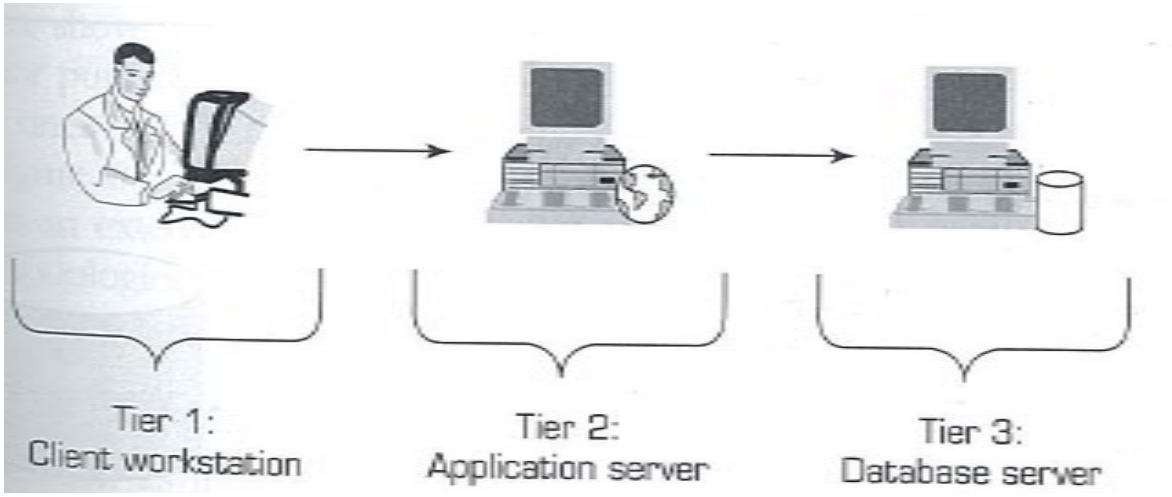

Figure 3.2 : Three-tier data warehouse architecture (Turban, 2007)

In a two-tier architecture, the DSS engine physically runs on the same hartware platform as the data warehouse. Therefore, it is more economical than the three-tier structure. The two-tier architecture can have performance problems for large data warehouses that work with data-intensive approach, maintaining that one solution is beter than the other, despite the organization's circumstances and unique needs. To further complicate these architectural decisions, many consultansts and software vendors focus on one portion of the architecture, therefore limiting their capacity and motivation to asist an organization through the options based on its needs.

Data warehousing and the Internet are two key Technologies that offer important solutions for managing corporate data. The integration of these two Technologies produces Web-based data warehousing. On the client side, the user needs an Internet

connection and a Web browser through the familiar graphical user interface (GUI). The Internet/intranet/extranet is the communication medium between clients and servers. On the server side, a Web server is used to manage the inflow and outflow of information between client and server. It is backed by both a data warehouse and an application server. Web-based data warehousing offers several compelling advantages, including ease of access, platform independence, an lower cost.

Web architectures for data warehousing are similar in structure to other data warehousing architectures, requiring a design choice for housing the Web data warehouse with the transaction server or as a separate server(s). Page-loading speed is an important consideration in designing Web-based applications; therefore, server capacity must be planned carefully.

3.4 EXTRACTION, TRANSFORMATION AND LOAD (ETL)

At the heart of the technical side of the data warehousing process is **extraction, transformation, and load (ETL).** The ETL process ian integral component in any data-centric Project. IT managers are often faced with challenges because the ETL process typically consumes 70 percent of the time in a data-centric Project. (Turban, 2007)

The ETL process consists of extraction (i.e., reading data from one or more databases), transformation (i.e., converting the extracted data from its previous form into the form in which it needs to be so that it can be placed into a data warehouse or simply another database), and load (i.e., putting the data into the data warehouse). Transformation occurs by using rules or lookup tables or by combining the data with other data.

ETL is extremely important for data integration as well as for data warehousing. The purpose of the ETL process is to load the warehouse with integrated and cleansed data. The data used in ETL processes can come from any source: a mainframe application, an ERP application, a CRM tool, a flat file, an Excell spreadsheet, or even a message queue.

The Figure 3.3 shows the ETL process.

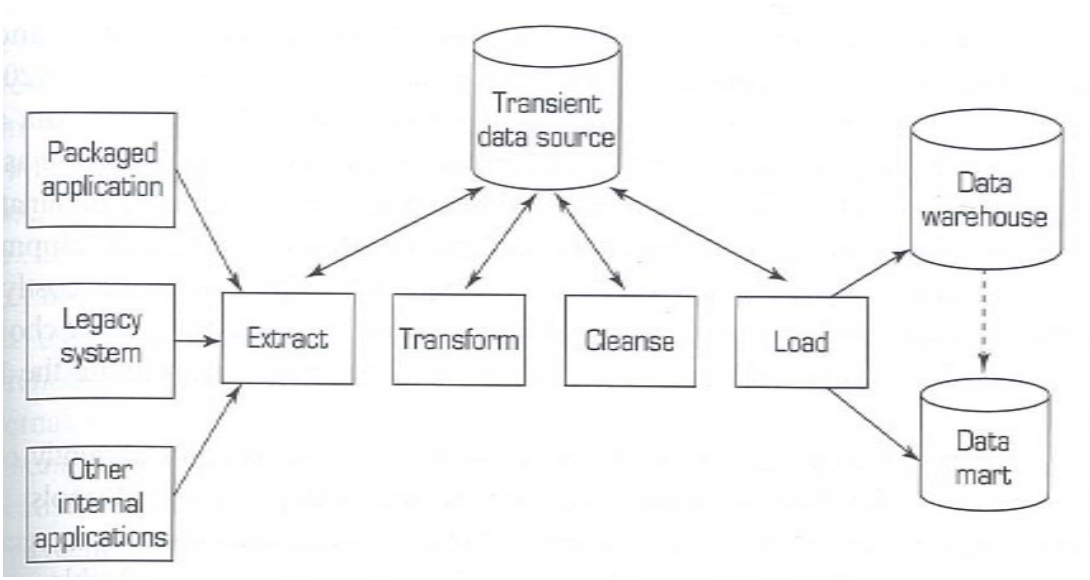

Figure 3.3 : ETL process (Turban, 2007)

The process of migrating data to a data warehouse involves the extraction of data from all relevant sources. Data sources may consist of files extracted from OLTP databases, spreadsheets, personal databases (e.g., Microsoft Access), or external files. Typically, all the input files are written to a set of staging tables, which are designed to facilitate the load process. A data warehouse contains numerous business rules that define such things as how the data will be used, summarization rules, standardization of encoded attributes, and calculation rules. Any data quality issues pertaining to the source files need to be corrected before the data are loaded into the data warehouse. One of the benefits of a well-designed data warehouse is that these rules can be stored in a metadata repository and applied to the data warehouse centrally. This differs from an OLTP approach, which typically has data and business rules scattered throughout the system. The process of loading data into a data warehouse can be performed either through data transformation tools that provide a GUI to aid in the development and maintenance of business rules or through more traditional methods, such as developing

programs or utilities to load the data warehouse, using programming languages such as PL/SQL, C++, or .NET Framework languages.

Global competitive pressures, demand for return on investment (ROI), management and investor inquiry, and government regulations are forcing business managers to rethink how they integrate and manage their businesses. A decision maker typically needs access to multiple sources of data that must be integrated. Before data warehouses, data marts, and BI software, providing access to data sources was a major, laborious process. Even with modern Web-based data management tools, recognizing what data to access and providing them to the decision maker are nontrivial tasks that require database specialists. As data warehouses grow in size, the issues of integrating data grow as well.

The business analysis needs continue to evolve. Mergers and acquisitions, regulatory requirements, and the introduction of nes channels can drive changes in BI requirements. In addition to historical, cleansed, consolidated, and point-in-time data, business users increasingly demand access to real-time, unstructured, and/ or remote data. And everything must be integrated with the contents of an existing data warehouse (Devlin, 2003). Many integration projects involve enterprise-wide systems. Properly integrating data from various databases and other disparte sources is difficult. But when it is not done properly, it can lead to disaster in enterprise-wide systems such as CRM, ERP, and supplyüchain Projects. (Turban, Efraim, 2007)

3.5 DATA WAREHOUSE DATABASE DESIGN

There are two basic models for database design: The relational model and the multidimensional model. The relational model is widely considered to be the “Inmon” approach, while the multidimensional model is considered to be the “Kimball” approach to design for the data warehouse. Both approaches have their advantages and disadvantages. (Khan, 2003).

3.5.1 The Relational Model

The relational approach to database design begins with the organization of data into a table. Different columns are in each row of the table. Table 3.1 shows a simple table.

column a
column b
column c
column d
column e
column f

Table 3.1 : A simple table

The relational table can have different properties. The columns of data have different physical characteristics. Different columns can be indexed and can act as identifiers. Certain columns may be null uupon implementation. The columns are all defined in terms of a data definition language (DDL) statement.

The relational approach to database design has been around since the 1970s and is well established through the relational implementation of Technologies such as IBM's DB2, Oracle's Oracle DBMS product, and Teradata's DBMS product, among others. Relational technology uses keys and foreign keys to establish relationships between rows of data. Relational technology carries with it the structured query language (SQL), which is widely used as an interface language from program to data.

Figure 3.4 shows a classical relational database design.

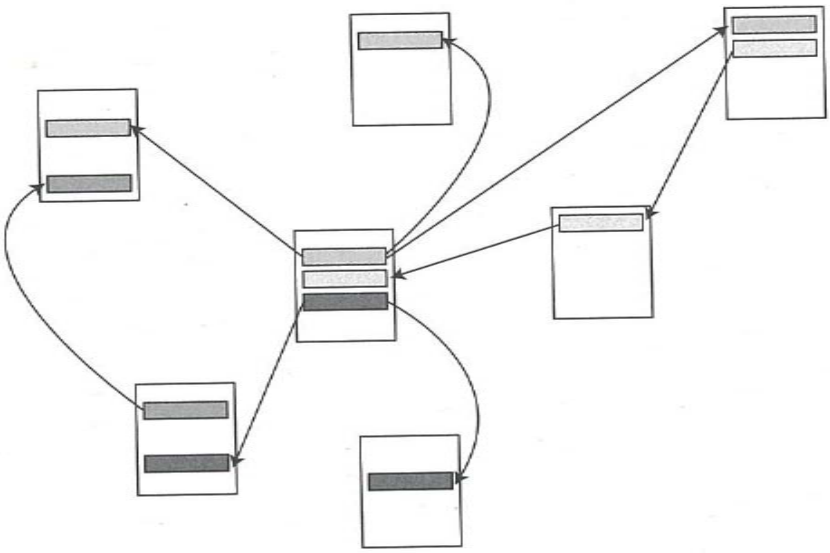

Figure 3.4: Classical relational database design (Inmon, 2005)

Figure 3.5 shows that there are different tables, and the tables are conneced by means of a series of key-foreign key relationships. The key-foreign key relationship is a basic relationship where an identical unit of data resides in both tables. (Inmon, 2005)

The data in the relational model exists in a form that is termed the "normalized" level. Normalization of data implies that the database design has caused the data to be broken down into a very low level of granularity. Data in a normalized form exists in an insular mode where data relationships within the table are very disciplined. When normalized, the data inside a table has a relationship with only other data that resides in the table. Normalization is said to typically exist at three levels- first normal form, second normal form, and third normal form.

The value of the relational model for database design for the data warehouse is that there is discipline in the way the database design has been built, clarity of the meaning,

and use of the detailed level of normalized data. In other words, the relational model produces a design fort he data warehouse that is very flexible. Database based on the desgin can be looked at first one way, and then another when the design has been based on the relational model. Data elements can be shaped and reshaped in many different ways. Flexibility, then, is the great strength of the relational model. Versatility is the second great strength. Because the detailed data is colected and can be combined, many different views of the data can be supported when the design fort he data warehouse is based on the relational model.

3.5.2 The Multidimensional Model

The other database design approach for the building of a data warehouse that is commonly considered is termed the multidimensional approach. The multidimensional approach is also sometimes called the star join approach. The multidimensional approach has been championed by Dr. Ralph Kimball. At te center of the multidimensional approach to database design there is the star join, as shown in Figure 3.5

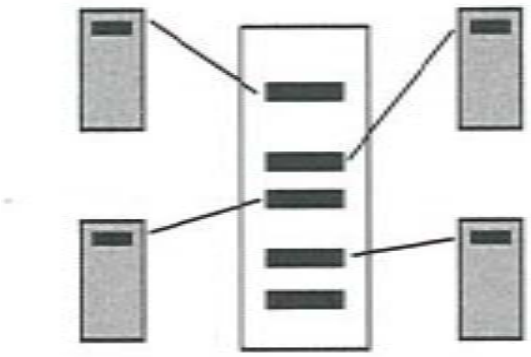

Figure 3.5 : Star join design (Inmon, 2005)

In opposition to relational databases, multi-dimensional databases have more than two dimensions.

Many data warahouse queries are multi-dimensional, which use multiple criteria against multiple columns, because the two-dimensional view of data limits the type of analysis that can be performed. Two-dimensional views cannot support the requirement to understand the relationship between multidimensions. In a relational database, analyzing multiple dimensions would require the setup of a series of tables. These tables could first be joined and then accessed through complex SQL code in order to analyze the cost trends over time. (Khan, 2003).

The need for joins, which are not difficult for programmers to implement, forces users to consider the dta structure. Multi-dimensional analysis over comes this limitation by accessing data throug more than one dimension or column (criteria). For example, it can enable analysis of sales by shipment by region over time.

The dimensional model overcomes the limitations of relational daabases, which are organized in a two-dimensional format. The dimensional model is based on a structure organized by dimensions, such as cargo transactions or geography, and is represented by a multi-dimensional array or cube. This model, which overcomes the two dimension limitation of relational databases, provides an intutive way of organizing and selecting data for querying and analysis. A multi-dimensional model:

- Is representative of the company's business model
- Provides a view that is business rather than technical; users can concentrate on the business instead of the tool
- Enables slicing and dicing, which provides the ability to analyze data using different scenarios such as sales by shipment, region, and period.
- Permits data to be easily analyzed across any dimension and any level of aggregation
- Is flexible and permits powerful analytical processing.

Multi-dimensional Structure's Concepts

Dimensions: A dimension represents an attribute such as cargo, region, or time. All data warehouse have one common dimension-time. A spreadsheet is the simplest example of a two-dimensional model. The spreadsheet row and column names are the "dimensions" while the numeric data in it are the "facts". A time dimension can include all months, quaarters, years, etc., while a geography dimension can include all countries, regions, and cities. A dimension acts like an index for identifying values in a multi-dimensional array. If the number of dimensions used is increased, greater is the level of detail that can be queried.

If a single member is selected from all dimensions, then a single cell is defined. A three-dimensional model is represented as a cubic structure in which each dimension forms a slide of the cube. In a dimensional model, data is organized according to a business user's perspective with common dimensions being time, region, cargo services, distribution or sales, and budget.

Facts: The values in the array in a dimensional model, which change over time, are called facts. Examples of facts, which are used to measure poerformance, include sales, units sold, costs, and shipments. Fact tables, which are the focus of dimensional queries, contain two thpes of fields.

- Fields that store the foreign key which connects each fact to the appropriate value in each dimension
- Fields that store individual facts suc has proce, quantity, salary, etc.

Characteristics of Fact and Dimension Tables

The following are the defining characteristic of facts and dimensions:

Fact table characteristics:

- Fact table consists of multiple columns and a large number of rows
  Is the primary table which contains the numeric data-measurements such as price, salary, volumes.
- Holds the "real" quantitative data-the data being queried; typically holds atomic and aggregate data such as the number packs sold with the cargos.

- Fact table contains all of the attributes to be measured.
- Fact table row corresponds to a measurement
- Measurement takes place at the intersection of all the dimensions such as month, product, and region
- Fact represents a business measure; fact attributes contain measureable numeric values (which are normally additive)
- Numeric measures are restricted to fact tables
- Facts can be operated upon(summed, averaged, aggregated, etc.)

Dimension table characteristics:

- Reflects business dimensions such as product, region, and distribution channel
- Contains a primary key that conects it to the fact table
- Dimensional attributes provide links between the fact table and its associated dimension tables
- Contains descriptive data reflecting business dimensions; dimensional attributes provide description of each row in the fact table
- Groups descriptive attributes about the facts; dimension table has many attribute fields; each field describes individual characteristics of the dimension; for example, attributes of unit dimension could be description of group, type, etc.
- Are used to guide the selection of rows from the fact table
- Dimensions permit categorization of transactions; example, customer dimension can be used to analyze procurement by location, frequency, etc.
- Ables are smaller as they have fewer number of rows
- Tables are de-normalied but that does not increase storage significantly as the dimension tables are very small compared to the fact table

Multi-dimensional Data Warehouse Database Designs

Star Schema

The start schema design, which is commonly used for designing data warehouse databases, supports analytical processing. It takes its name from the star-like

arrangement of entities. The star schema is the design most frequently used to implement a multi-dimensional model in a relational database. Its structure consistes of a central fact table with keys to many dimension tables (Figure 3.6)

The following characteristics are associated with a star schema:

- It contains two types of tables: Fact (or major) and Dimension (or minor)
- One dimension represents one table
- Dimension tables are de-normalized
- Dimension tables are linked to the fact table through unique keys (one per dimension table)
- Every dimension key uniquely identifies a row in the dimension table associated with it
- A fact table's specific row is uniquely identified by the dimension keys
- Uses many ERD components such as entities, attributes, cardinality, primary keys, and relationships connectors.

The star schema design has many advantages. It favors de-normalization for optimizing speed. Due to de-normalizition of the time dimension, a significant reduction occurs in the number of tables that need to be joined when time-based queries are executed. A star schema's poerformance is good because one large table needs to be joined with a few small tables, resulting in a fast response time. The star schema reflects how business users view data, makes metadata navigation easier for both programmers and end-users, and permits more versatility in the selection of front-end tools.

Snowflake Schema

If a dimension table has subcategories or more than one level of dimension tables, and more efficient Access is required, a snowflake schema can be used. The snowflake schema, which is derived from the star schema, adds a hierarchical structure to the dimension tables (Figure 3.7). It is more normalized and complex.

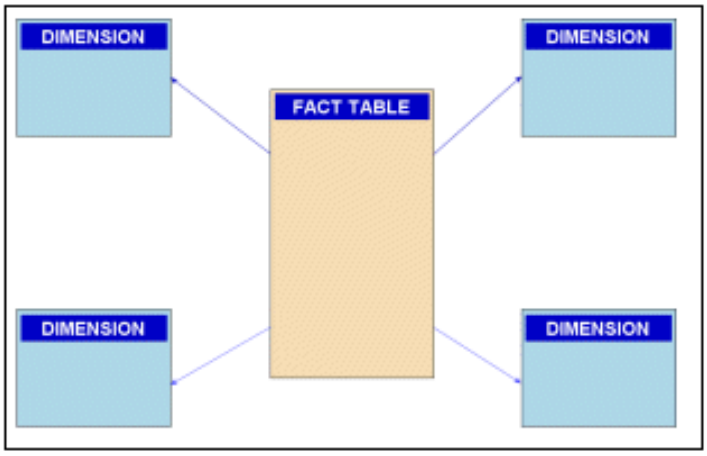

Figure 3.6 : Star Schema

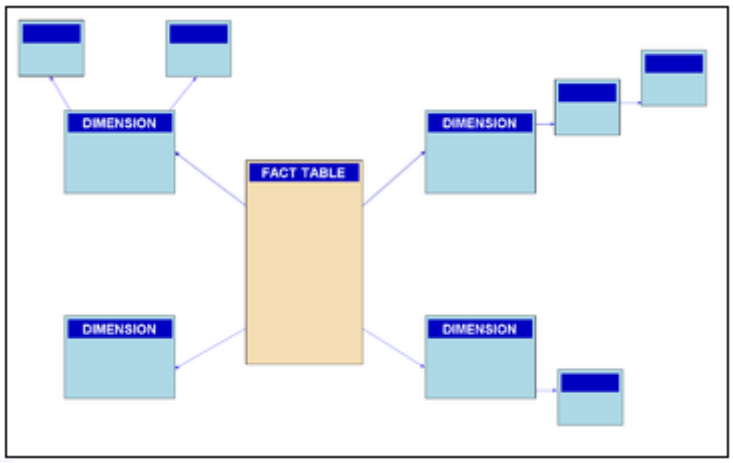

Figure 3.7 : Snowflake Schema

OLAP

Online analytical processing (OLAP), is an analytical technique that combines data access tools with an analytical database engine. In contrast to the simple rows and columns structure of relational databases, upon which most data warehouses are built, OLAP uses a multi-dimensional view of data such as sales by cargo services, quarters, and cargo types. OLAP, which Works on data aggregations, uses calculations and

transformations to perform its analytical tasks. (Khan, 2003). There are two basic types of OLAP system architectures:

- Multi-dimensional OLAP (MOLAP)
- Relational OLAP (ROLAP)

OLAP Database Server

An OLAP server stores data as well as the relationships between the data. It is optimized for ad hoc query processing and data maniulation. An OLAP server is designed to work with multi-dimensional data structures, which can be visualized as cubes of data (and cubes within cubes), with the following characteristics:

- A cell is a single point in a cube
- Each data item is located, and accessed, based on the intersection of the dimensions defining it
- Each side of a cube is a dimension that represents an attribute or category such as units, region, customer type, or time period.
- Each cell contains aggregated data that relates the elements along each dimension
- Using the dimension numbers that define them, data items can be easily located and accessed
- An intermediate server can be used to store pre-calculations

An OLAP server's key characteristic is its calculation engine. An OLAP server can extract data in real-time from relational or other databases and, when required, manipulate it. However, the more common and preferred method is to physically store the data on the OLAP server in multi-dimensional format. A database which stores data in multi-dimensional format is known as a multi-dimensional database (MDDB).

Benefits and Features of OLAP

OLAP technology enables decision makers to Access data quickly, efficiently, interactively, and in innovative ways without first having to understand the data structure or technical details. The data, which is presented in dimensions as business users view it, can be queried and analyzed using different views. Compared to data

warehouses based on relational database technology, OLAP systems have an additional feature-the ability to perform "what if" analysis, a powerfl tool that can simulate the effect of decisions.

The following are basic benefits of OLAP technology:

- Enable users to identify key trends and factors driving their businesses
- Ability to perform complex calculations and trend analysis
- Ability to manipulate data with many inter-relationships
- Insulate users from SQL language and he relational model
- Improve query performance; massive amonts of data can be analyzed repidly
- Improve scalability
- Support a wide range of tools
- Automate maintenance of İndexes and summaries
- Decrease demand for reports from IT
- Fast deployment
- Application in a wide range of applications such as forecasting, profitability analysis, customer analysis, budgeting, and marketing analysis
- Increase productivity of individuals and organizations

These are the desired OLAP features and characteristics:

- User perspective: data should be transparent to the users
- Ease of use
- Intuitive data manipulation
- Easy and fast deployment
- Seamless presentation of historical, projected, and derived data
- Reasonable cost
- Cost-effective maintenance
- Ability to perform operations against single or multiple dimensions (aggregate, summarize, and derive data)
- Powerful calculation capabilities
- Support for statistical and analytical functions

- Support more than simple aggregation or roll-ups such as share calculations and allocations
- Support for large data sets and unlimited dimensions and aggregation levels
- Time intelligence which suppors analysis such as year-to-date and perido-over-period
- Provide serce and concurrent access to data
- Consistent and fast query performance
- Flexible reporting; consistent reporting performance
- Integration with desktop tools
- Scalability-large data volumes as well as the number of concurrent users
- Permit data to be read while updates are occurring

MOLAP versus ROLAP

In multi-dimensional OLAP (MOLAP), data is stored in special OLAP database server, after being extracted from various sources, in pre-aggregated cubic format. This data remains static until an extract from the source system(s) adds more data to it. In contrast to this approach, relational OLAP (ROLAP), does not use an intermediate server because it can work directly against the relational database. Consequently, it can perform analysis on the fly.

MOLAP performs well with 10 or fewer dimensions while ROLAP can scale considerably higher. ROLAP is not restricted by the number of dimensions, type or number of users, database size, or complexity of analysis. It can perform ad hoc queries and aggregate data much faster-even with constantly changing and a much larger amount of data. Another ROLAP advantage is that it can leverage paralel scalable relational databases. The disadvantages of ROLAP are that it has limited scalability, places a heavy load on the server, and is expensive to maintain.

MOLAP, which starts seeing performance degradation at about 30-50 GB of data or 10 dimensions, is more suitable for financial applications where the data can be broken down and is smaller. ROLAP is more suitable for applications where a huge amount of data needs to be analyzed, such as marketing and point-of-sale.

Hybrid OLAP (HOLAP)

Hybrid OLAP (HOLAP) combines the features of ROLAP and MOLAP. It takes advantage of the superior processing of MOLAP with the ability of ROLAP to work with greater data volumes. HOLAP stores data in both a relational database and multi-dimensional database (MDDB). Either database can be used depending on the type of processing required-data processing or ad hoc querying. In HOLAP, the aggregations are stored using a MOLAP strategy while the source data, which is far greater in volume, is stored using a ROLAP strategy. The result is that the least storage is used while enabling very fast processing.

The hybrid OLAP system combines the performance and fuctionality of the MDDB with the ability to access detail data, which provides greater value to some categories of users. However, HOLAP implementations are typically supported by a single vendor's databases and, also, are fairly complex to deploy and maintain. Additionally, they can be somewhat restrictive in terms of their mobility.

4. THE DSS NEED OF THE TRANSPORTATION COMPANY

4.1 TRANPORTATION COMPANY ANALYSIS

Aras Cargo is one of the two biggest transportation companies of Turkey. The company serves 6 million people, institutions and companies of every month with its 20 district offices, 27 transfer stations, 754 contact offices, a fleet of 2500 vehicles and an expert team of 7700 people. Take services to over 1500 residential units in all towns and villages across the country, Aras Cargo is also expanding its service area every day with mobile services operating in about 800 population centers.

The avarage number of freight that the company makes in one day is 220,000. These shipments consist of 700,000 pieces of cargo. All pieces are nearly 10 cargo operational transactions. This makes 2.1 billion operational transactions. Through these processes, financial transactions and sales data have also emerged. With all processes, over 4 billion transaction is created in one year. These all transactions make two tera bytes of data in one year.

Organization's giro increased from 65 million dolar to 400 million dolar in 8 year. With this growth, company needs more management analysis and reports.

4.2 TRANPORTATION COMPANY NEEDS

With the growth of the compant, the need for institutionalization becomes very important subject. One of the first necessity of the institutionalization for a company is monitoring and analyzing the business.

These are the needs of the company:

- Organization needs operational giro analysis, shipment analysis, cargo transaction's analysis, financial analysis. In all these analysis, it's needed to see all parameters

about transportation activities. But the reports of the operational systems give statical results with the restricted parameters. Unfortunately existing reports can return just for small period of time.

- One of the most important subjects for the firm is lack of monitoring and the analysis the real effects of unit's operation to the company. The regions and the branchs are assessed by the giro amount without other parameters which they did in a period of time.
- In fact, their effect to the firm is also dealing with other parameters, such as the calculation of the cost for per kilogram of cargo that was carried. Another important subject about giro reports is the distance parameter. The firm can't analyze the giro according to the distance.
- For financial analysis, there's need for calculation about activity base costing. The financial department's data and operational department's data can't be used together because of the size of the data.
- In the sales department, sales managers can't analyze the sales according to 'products' and 'month period of time' parameters. In the term of the rasing of the prices, they can't see sales parameters in giro amont. Because of that, sales department can't decide the rate of price's rise analytically.
- In the operational department, head quarter can't calculate the exact performance of the units with using many parameters. The shipments which are processed in the hub units, can't be analyzed with the many parameters effectively because of the size of the data.
- For making operational decisions about opening a new unit or a hub, there's need to have the geographically shipment data. With this data, the route optimization will also be enable.
- The company needs to estimate the next month's sales and wants to determine sales strategies.

4.3 PROBLEMS OF THE COMPANY

Company has got reporting server in the automation systems. But users can't run reports for more than 6 month's of data and can't make trend analysis.

When the queries are run that uses the operational system's databases for running the analyzing reports, the operational systems have also effected. The daily processes slowed down and sometimes the reporting queries blocked the databases activities. It is understood that the reports which need big size of data, must be run on a separate database which is different from operational systems' databases.

For being a solution to all these analysis and reporting requirements, developing a decision support system that uses a separate database which is designed just for analysis, is decided.

5. WAREHOUSE MODEL FOR THE DECISION SUPPORT SYSTEM

5.1 CHOOSING AND DESIGNING THE DATA WAREHOUSE MODEL

In the main automation system of the company, there are 1085 tables which are using different purposes. Some of the tables are used separately from the main database.

From 4 database, different kinds of data will be transformed to data warehouse system. The operational system, which will be used as a source system of the data warehouse system, has the data of operational transactions, financial transactions etc. These transactions are connected between each other. But the financial data and the operational data is not suitable to use in one multidimensional structure. So it's understood that one multidimensional structure can't be a solution for DSS.

It is seen that there will be need to analyze with using many parameters. The decision makers will also need to make trend analysis for a long period of time, such as for 1 or 2 years. The relational data warehouse model won't be the best solution because of the necessity to the high performance.

When I analyzed the variety of the business report requests and the different kinds of data of the OLTP system, I decided the Hybrit OLAP model for using in the DSS.

For Hybrit OLAP model, I analyzed the Fact tables and Dimensions tables. I joined the base tables which are connected with the functional relations and created the fact tables.

An example of Cargo data of OLTP system, the relations are seen in Figure 5.1

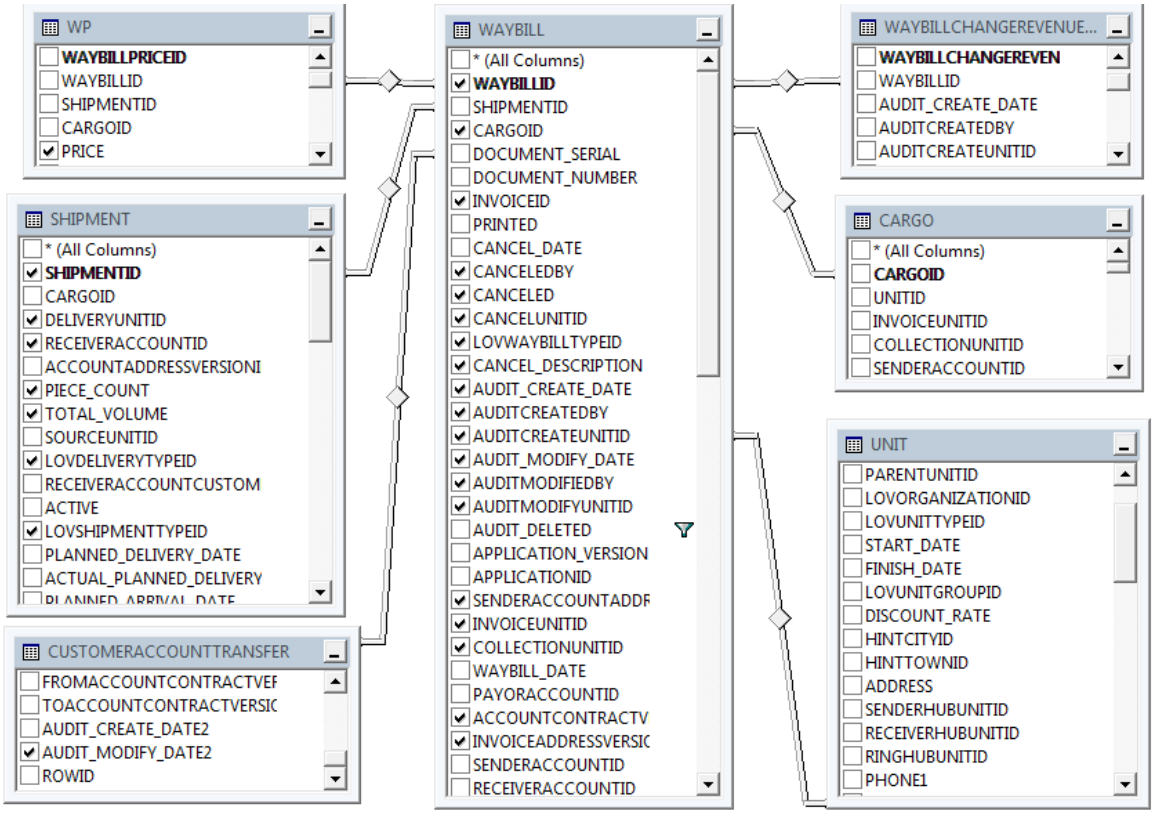

Figure 5.1 : Cargo data of the OLTP System

With the ETL processes, I joined the tables which are related with each other, and transformed the Cargo data as Cargo_fact table to the Data Warehouse System. The SQL for creating the cargo fact table of the Data Warehouse System is seen in Figure 5.2.

```
select
trunc(waybill_Date) OPERATION_DATE, w.auditcreateunitid SOURCEUNITID,
W.DOCUMENT_SERIAL||W.DOCUMENT_NUMBER SERIAL_NUMBER, S.SHIPMENT_CODE SHIPMENT_CODE, S.SENDERACCOUNTID,
S.RECEIVERACCOUNTID, W.WAYBILLID, S.SHIPMENTID, W.CARGOID, W.CAMPAIGNCONDITIONID,
W.ACCOUNTCONTRACTVERSIONID, decode(nvl(W.WORLDWIDE,0),0,'YURTICI','YURTDISI') YI_YD, S.PIECE_COUNT
PIECE_COUNT, S.TOTAL_VOLUME TOTAL_VOLUME, C.LOVPAYORTYPEID, S.LOVPACKTYPEID, W.SERVICEID
PRIMARYSERVICEID, WP.PRICE TOTAL_PRICE, WP.INVOICE_PRICE TOTAL_INVOICE_PRICE,
TO_CHAR(TO_dATE(waybill_Date, 'DD.MM.RRRR'),'RRRR')|| TO_CHAR(TO_dATE (waybill_Date,
'DD.MM.RRRR'),'MM') || TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RRRR'),'DD')  OPERATIONDATEID,
trunc(S.ACTUAL_PLANNED_DELIVERY_DATE) ACTUAL_PLANNED_DELIVERY_DATE, trunc(S.PLANNED_ARRIVAL_DATE)
PLANNED_ARRIVAL_DATE, trunc(S.FIRST_PIECE_ARRIVAL_DATE) FIRST_PIECE_ARRIVAL_DATE,
trunc(S.LAST_PIECE_ARRIVAL_DATE) LAST_PIECE_ARRIVAL_DATE, S.LOVUNITDISTANCETYPEID, S.DURATION,
S.MOBILE, S.MANIFESTUNITID, S.DESCRIPTION, S.LOVSHIPMENTSTATUSID, S.INTEGRATION_CODE,
S.CONTENTS_DESCRIPTION, trunc(S.DELIVER_DATE) DELIVERY_DATE, S.SENDERACCOUNTNAME,
S.SENDERACCOUNTADDRESSNAME, S.RECEIVERACCOUNTNAME, S.RECEIVERACCOUNTADDRESSNAME, S.DELIVERYUNITID,
S.LOVDELIVERYTYPEID, S.RESPONSIBLEUNITID, trunc(S.SHIPMENT_DATE) SHIPMENT_DATE,
S.SHIPMENTDELIVEREDID, S.LOVDELIVERYFAILUREREASONID, S.PARENTDELIVERYUNITID, S.PARENTSOURCEUNITID,
trunc(W.CANCEL_DATE) CANCEL_DATE, W.CANCELEDBY, W.CANCELED, W.AUDIT_CREATE_DATE AUDIT_CREATE_DATE,
W.AUDITCREATEUNITID AUDITCREATEUNITID, W.INVOICEID, W.CANCELUNITID, W.LOVWAYBILLTYPEID,
W.CANCEL_DESCRIPTION, W.AUDITCREATEDBY, W.AUDITMODIFIEDBY, W.AUDITMODIFYUNITID,
W.SENDERACCOUNTADDRESSVERSIONID, W.INVOICEUNITID, W.COLLECTIONUNITID, W.INVOICEADDRESSVERSIONID,
W.WORLDWIDE, W.LOVDOCUMENTPRINTSTATUSID, W.DIFFINVOICEID, W.PAYORACCOUNTCUSTOMERID, W.RETURNED,
W.PUANTUMBONUS, W.PUANTUMCANCEL, W.PUANTUMCANCELED, W.PUANTUMCARDNO, W.PUANTUMCARDOWNER,
W.PUANTUMWEB, trunc(W.DUE_DATE) DUE_DATE, W.PAYORACCOUNTNAME, W.INVOICEADDRESSID, W.ACCCONTRACTID,
W.RECEIVERADDRESSVERSIONID, W.REFCODE, W.TRADING_WAYBILL_NUMBER, C.TRADING_GOODS,
C.RESPONSIBILITY_DOCUMENT, C.RECEIVEEMPLOYEEID, C.MEASUREEMPLOYEEID, C.CONTACT_NAME,
C.LOVIDENTITYTYPEID, C.CONTACT_IDENTITY_OFFICE, CONTACT_IDENTITY_NUMBER,
C.PAYMENTACCOUNTCONTRACTVERSID, C.PRICELISTID, C.CARGOCOLLECTID, C.LOVCARGOSTATUSID, C.PARTY_CODE,
C.LOVPARTNERID, C.CARGONOTICEID, trunc(C.TRADING_DATE) TRADING_DATE, C.FOR_WORLDWIDE, C.WWCARGO_VALUE
,TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(S.PLANNE
D_DELIVERY_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'D
D')  PLANNEDDELIVERYDATEID
,TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(S
.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_D
ATE),'DD.MM.RRRR'),'DD')  ACTUALPLANNEDDELIVERYDATEID
,TO_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(S.PLANNED
_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
PLANNEDARRIVALDATEID
,TO_CHAR(TO_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(S.DELIVER_DATE),'
DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'DD')  DELIVERYDATEID
,TO_CHAR(TO_dATE(trunc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(S.SHIPMENT_DATE)
,'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'DD')  SHIPMENTDATEID
,TO_CHAR(TO_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(W.CANCEL_DATE),'DD
.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'DD')  CANCELDATEID
,TO_CHAR(TO_dATE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(W.DUE_DATE),'DD.MM.RR
RR'),'MM')||TO_CHAR(TO_dATE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'DD')  DUEDATEID
,TO_CHAR(TO_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(trunc(C.TRADING_DATE),'
DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'DD')  TRADINGDATEID
from
waybill w
inner join waybillprice wp on w.waybillid=WP.WAYBILLID
inner join shipment s on  s.shipmentid=w.shipmentid
inner join cargo c on    c.cargoid=w.cargoid
inner join UNIT U on      U.UNITID=W.AUDITCREATEUNITID
inner join customeraccounttransfer cat on  CAT.WAYBILLID=W.WAYBILLID
inner join waybillchangerevenueunit wcr on   wcr.WAYBILLID=W.WAYBILLID
```

Figure 5.2 : The SQL code for creating the cargo fact table

I used the lookup tables of the OLPT system as dimensions tables in the data warehouse systems.

With the cargo data, I also transport the lookup tables deals with cargo table, such as ‘region of the cargo’, ‘the product of the cargo’ and ‘month period of time’ lookups. With this structure I’ve got the multidimensional structure.

An example of cargo dimensions are seen Figure 5.3.

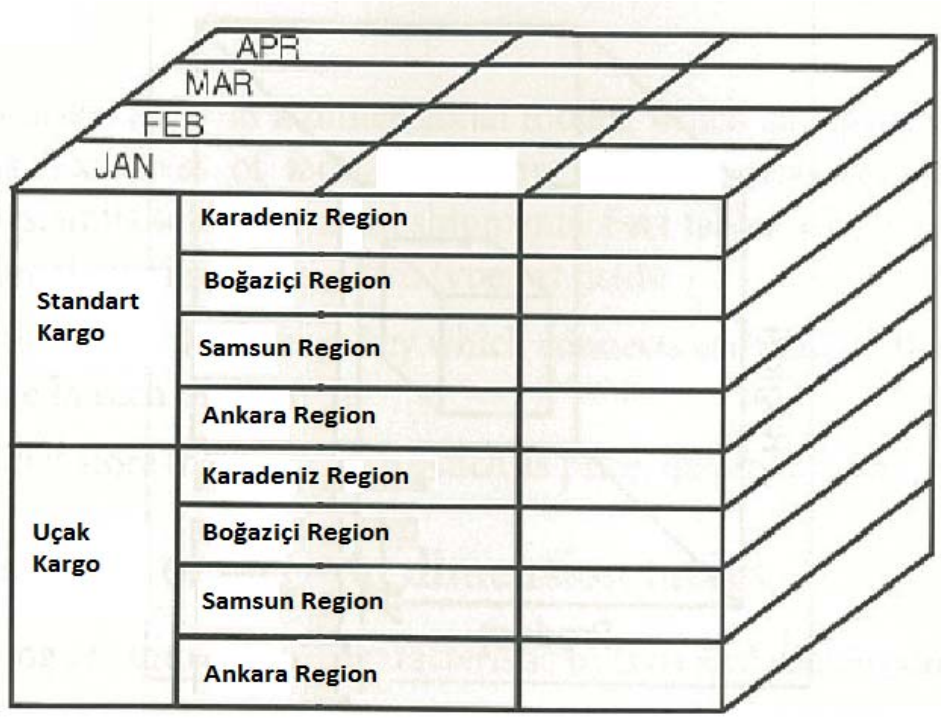

Figure 5.3 : Cargo's dimensions

5.2 ETL PROCESS DEVELOPMENT

The data warehouse system will export data from 4 different source databases. Three of them are the automation systems, one of them is ERP system.

For ETL processes, I programmed a transformation system. The data is transferred with this system from source systems to warehouse system. I used the ETL structure as shown in Figure 5.4

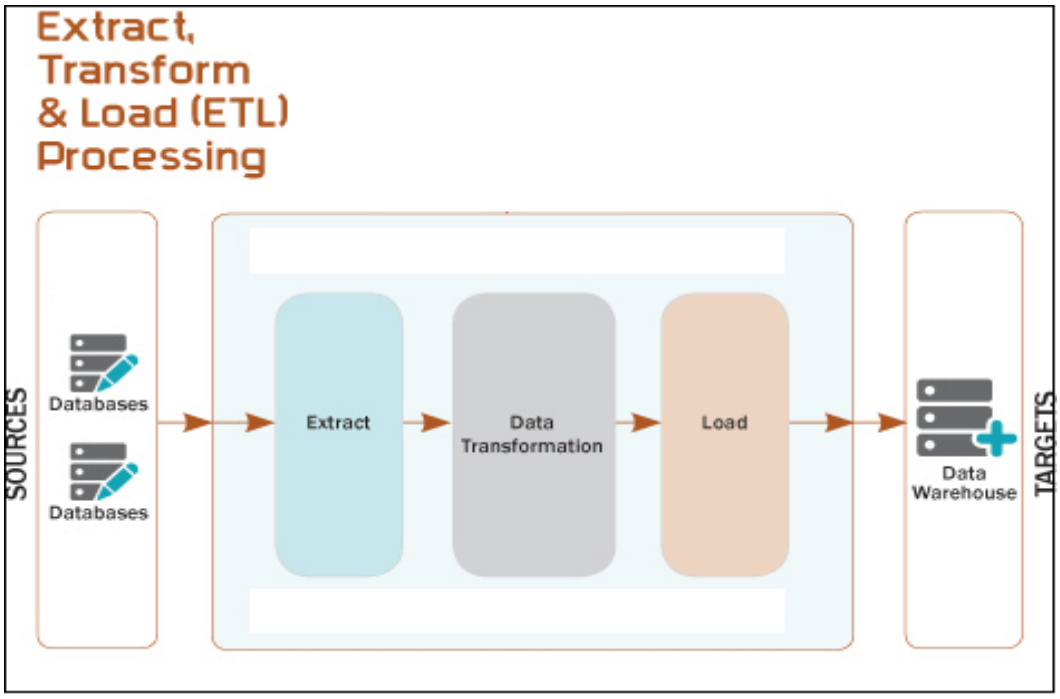

Figure 5.4 : ETL structure

In the 'Extract Phase' of the ETL, data is selected. These procedures run on the source databases. Extracting procedures prepare data and insert it to temp tables in the source databases. (Appendix A)

Extracting processes are based on catching the new or modified data. Because of source system's cargo data aren't created online in the transportation company, the creating and updating time information on the rows aren't suitable to use in ETL processes. To solve this problem, I added 'AFTER INSERT' and 'AFTER UPDATE' triggers to the source system's tables which will be transferred to the data warehouse system. Triggers are seen in Figure 5.5.

```
CREATE OR REPLACE TRIGGER ESASLIVE."Ins_WAYBILL" before insert
on WAYBILL

  for each row

begin

      :new.AUDIT_ MODIFY_DATE2:=CURRENT_DATE;

end;
/


CREATE OR REPLACE TRIGGER ESASLIVE."UPD_WAYBILL" BEFORE UPDATE
on  ESASLIVE.WAYBILL

REFERENCING NEW AS NEW OLD AS OLD

  for each row

begin

      :NEW.AUDIT_MODIFY_DATE2:=CURRENT_DATE;

end;
/
```

Figure 5.5 : Before insert and before update triggers

When inserting or updating a data on the source system's tables by operational systems, these triggers run and insert or update the same date column of the row.
Extracting procedure selects the data from the last time which it worked to the sysdate of the database with using the time logging table. After extracts the data, procedure add new time log row to the logging table.

With this methodology, ETL load the data also when creating new row or modifying in the source system's database tables. So, data which is created or updated is carried with the same temp table.

When the extracting procedure completes its processes, 'Transformation and Loading Procedure' begin its processes. Data is transferred and loaded with this procedure which runs on the data warehouse database. This procedure takes the data from the source database's temp table and inserts it to de warehouse temp tables which were prepared by the 'Extract Phase' before. After finishing the transformations of the data between temp tables, procedure use the data warehouse's temp table and merge the data to the warehouse database tables. (Appendix B) With merging method, it's prevented to insert the same data to the data warehouse system more than once. If it's the first time

that data is transported to the data warehouse system, it's inserted. If not, data in the data warehouse system is updated.

5.3 CHOOSING THE ANALYZING TOOL

For choosing a BI tool, first I analyzed the company's business intelligence strategy. Department's report requests showed that big amount of data would be used for analyzing. The DSS system would need high ability of a multidimensional structure. Firm managers would use the DSS for analyzing the data with using every dimension that the transportation sector has.

I defined other business critical selection criteria as self-service reporting. Self-service reporting ability is one of the most important subjects of the project. The end user managers will need to make their own report by theirselves. The IT department will stop using effort to make reports for other departments. The end users will be able to shape their reports with using the objects which has previously prepared for creating reports by the IT staff.

After the decision makers' needs were analyzed, we invited the vendors for a live demonstration of their solutions. With the Proof-of-concept (PoC), we tested the solution and got an idea of the functionality, connectivity, usability and performance of each BI tool.

	KEY REQUIREMENTS FOR COST REDUCTION	MICROSTRATEGY 9	ORACLE BI EE Plus 10g R4	IBM COGNOS 8.4	MICROSOFT
Minimizing Design Effort		YES	LIMITED	LIMITED	NO
	Dynamic Report Personalization	• Flexible, easy to use column and object prompts allow business users to choose from all reporting, analysis, and business logic objects to author their own reports at run time	No object or column prompts such as selection of attributes, metrics, and filters for report creation on-the-fly. This causes an unnecessary number of reports to be created and maintained for end users.	• No object or column prompts such as selection of attributes, metrics, and filters on-the-fly. This causes an unnecessary number of reports to be created and maintained for end users.	• Business users must ask IT to add or remove report objects; business users do not have the option to select any object from any data source at run time.
	Automatic Multi-source Drill Anywhere	Business users can automatically drill anywhere to any data source without IT hard coding.	No automatic drill anywhere; drilling across hierarchies requires IT hard coding the report destinations.	No automatic drill anywhere, drilling across hierarchies require IT hard coding to report destination.	Microsoft end users cannot automatically drill across data to any data source.
	Formatting over the Web	Easy What-You-See-Is-What-You-Get (WYSIWYG) formatting allows business users to format reports at runtime without IT support.	End users get dashboards with static versions of reports. Users cannot change the format, cannot pivot, cannot sort by, cannot edit reports on-the-fly.	Cognos end users get static reports only, can't change format, can't pivot, can't sort by, can edit reports on-the-fly.	End Users must wait for IT to add or remove subtotals or formatting such as background colors.
	One Repository of Reusable Business Logic	Report developers can reuse all existing business logic across the entire platform rather than spending time recreating business logic.	OBIEE Plus legacy of stand-alone products are still not fully integrated on a single repository of reusable business logic run on separate repositories.	Cognos legacy of stand-alone products are still not fully integrated on a single repository of reusable business logic	IT must recreate logic for each new report, increasing development time and potential for multiple versions of the truth.

Table 5.1 : The comparison of the Design Effort

Minimizing Deployment Effort	One Report Design Automatically Deploys to Any Interface	The same report design is automatically optimized for interacting through all interfaces including Web browsers, mobile devices, and Microsoft Office	Oracle Answers prompted reports can be used in Oracle Interactive Dashboards but the same report cannot be reused on BI Publisher. Oracle Interactive Dashboards are not supported in Oracle Delivers.	Cognos provides several formats from one report definition. However, its Flash support is significantly limite.	Microsoft BI is optimized for IE;developers need to hard code to display for other browsers (Firefox).
	Browser Agnostic Zero-footprint Web	Eliminates client installation costs and ensures application is automatically updated.	Oracle BI Briefing Books for offline dashboard viewing require the installation of a desktop client in the user's machine.	Cognos report designers that use its Report Studio have to use Microsoft Internet Explorer (IE)	Microsoft requires user to have permissions to download client plugins for Report Builder and for printing reports.
	Easy to Customize and Upgrade	Upgradable plug-ins instantly customize to corporate look and feel without coding effort.	OBIEE Plus does not offer a single code base for application customization. OBIEE Plus does not offer an Eclipse IDE plug-in or support for Flex Builder to speed up customization.	Cognos does not have any Eclipse IDE plug-in or support for Flex Builder to speed up customization and migration tasks.	Customizations require extensive IT coding.
	Automated Deployment	Automated life cycle management tool synchronizes objects across development, test, and production environments thus greatly reducing manual work associated with BI deployments.	OBIEE Plus does not offer a comprehensive tool for automated life cycle management and to consolidate and reconcile disparate departmental BI applications.	Cognos does not offer a comprehensive tool like MicroStrategy Object Manager for automated life cycle management	Microsoft requires IT staff to complete a manual publishing process before end users can access cubes or reports.
Minimizing Administration Effort	Single Server	A centralized server dramatically reduces administrative effort and complexity.	OBIEE Plus uses a number of disparate servers (e.g., Oracle BI Server, Oracle BI Presentations Services server).	Cognos BI uses a number of disparate servers (e.g., Cognos BI, Cognos Virtual View Manager, Applix TM1, and Celequest).	Microsoft requires an administrator to install and separately administer many servers.
	Single Point of Administration	Administrators need only create users and security settings once.	OBIEE Plus multiple legacy technologies and servers expose multiple administration points, increasing effort and complexity of deployments.	Cognos multiple platforms and servers expose multiple administration points, increasing effort and complexity of deployments.	Microsoft requires many user metadata, increasing administrator effort and increasing security risks.

Table 5.2 : The comparison of Deployment and Admin Effort

As a result of the surveys and POC's, it's seen that, the Microstrategy tool is the best choice for the company. The comparision of tools are seen on Table 5.1 and Table 5.2

5.4 BUILDING ANALYZING OBJECTS OF THE DATA WAREHOUSE

5.4.1 Creating Analytical Reporting Project

With Microstrategy BI tool, first I created a project which name is ABI. (Figure 5.6)

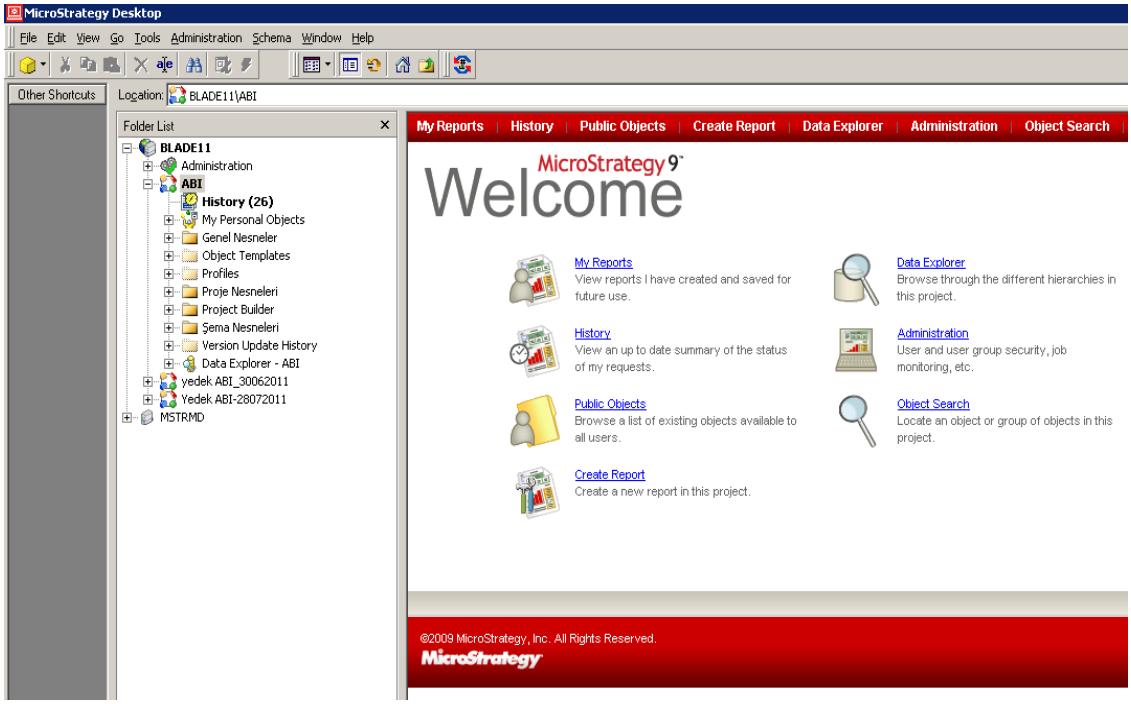

Figure 5.6 : Creating a project with BI tool

A project is where we build and store all schema objects and information you need to create application objects such as reports in the MicroStrategy environment, which together provide a flexible reporting environment. (MicroStrategy 2010).

A project:

- Determines the set of data warehouse tables to be used, and therefore the set of data available to be analyzed.

- Contains all schema objects used to interpret the data in those tables. Schema objects include facts, attributes, hierarchies, and so on. Schema objects are discussed in later chapters in this guide.
- Contains all reporting objects used to create reports and analyze the data. Reporting objects include metrics, filters, reports, and so on. Report objects are covered in the MicroStrategy Basic Reporting Guide and the MicroStrategy Advanced Reporting Guide.
- Defines the security scheme for the user community that accesses these objects. Security objects include security filters, security roles, privileges, access control, and so on.

After creating the project, I connected the project with the warehouse data source. With this connection, I added the tables and views which were warehouse's fact and dimension tables, to the project's warehouse catalog. (Figure 5.7).

For the first phase of the project, 296 tables (25 fact table and 271 dimension table) imported to the project.

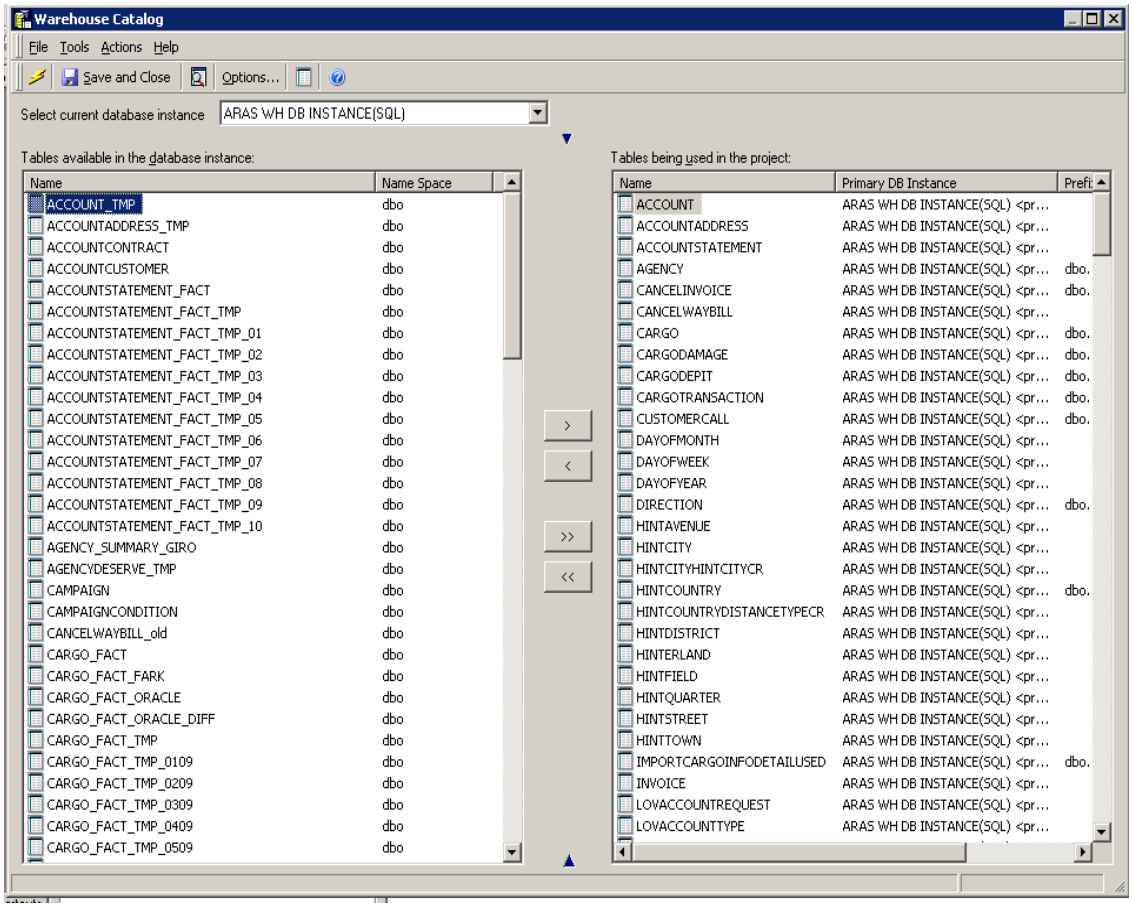

Figure 5.7 : Adding tables to the Warehouse Catalog

5.4.2 Defining Objects of the Project

After adding data warehouse tables to the project, I defined the objects of the analyzing project. These are Configuration Objects and Schema Objects.

Defining Configuration Objects:

Configuration objects: Objects that provide important information or governing parameters for connectivity, user privileges, and project administration. These objects are not used directly for reporting, but are created by a project architect or administrator to configure and govern the platform. (MicroStrategy 2010).

For the project, database instances, users, groups, security roles are included.

For using the DSS system, hierarchical user groups were defined which had headquarter privilages, region privilages, hub unit privilages and branch privilages. Besides of that hierarchy, also functional privilages were defined for user groups such as operational department privilages, sales department privilages, finance department, customer department privilage. Then I crossed these two kinds of privilage grouping. So, every different unit's different department will be able to analyze their own data that they can use. It's shown in the Figure 5.8.

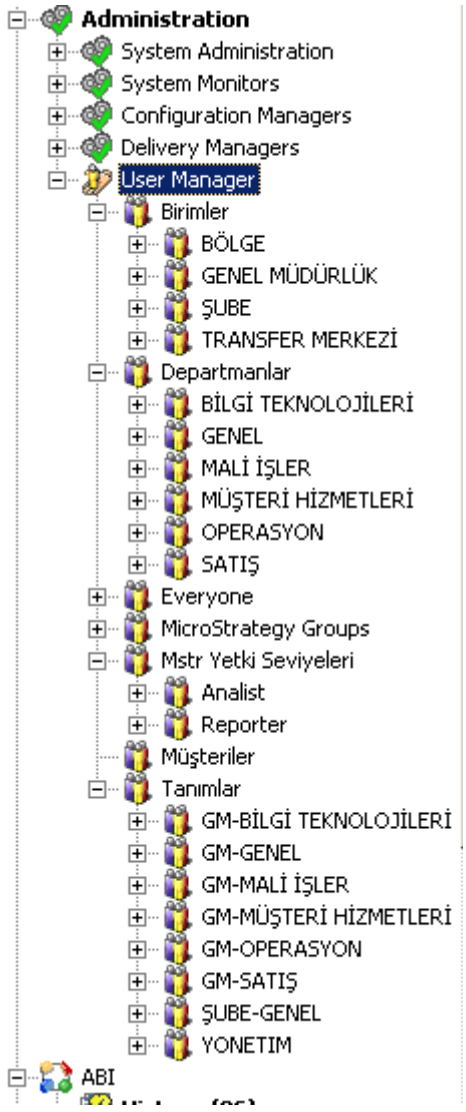

Figure 5.8 : Configuration objects of the transportation DSS system

Defining Schema Objects

Schema objects: Objects that are created in the application to correspond to database objects, such as tables, views, and columns. Schema objects include facts, attributes, metrics and other objects.

Defining Facts

Facts relate numeric data values from the data warehouse to the MicroStrategy reporting environment tool. Facts are used to create metrics, which are analytical calculations that are displayed on a report.

A fact has two common characteristics: it is numeric and it is aggregatable. Facts objects are created with the fact table's numeric column of the warehouse. (MicroStrategy 2010).

The structure of facts

As shown in the Figure 5.9, facts are made up of the following components:

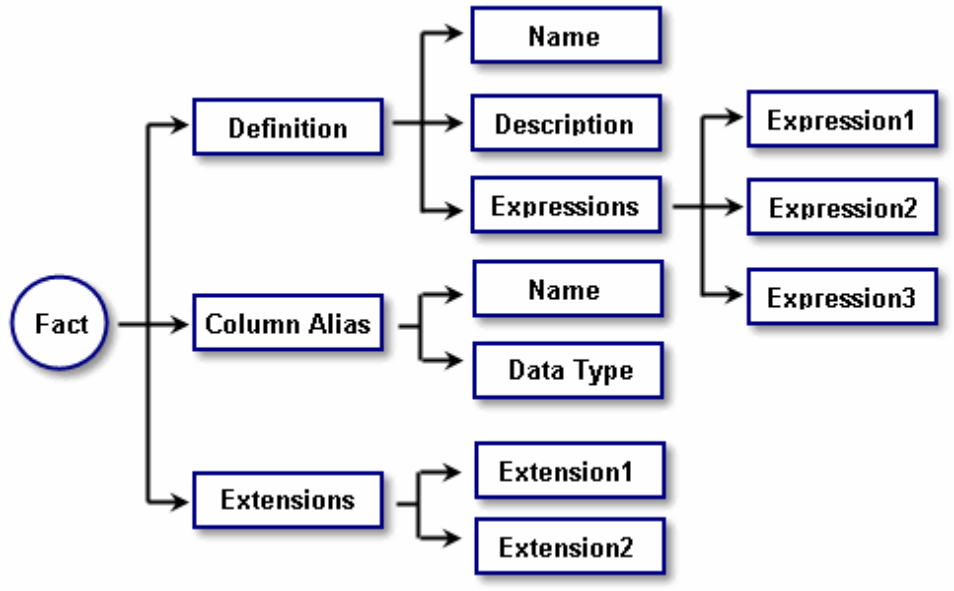

Figure 5.9 : The structure of facts

The **fact definition** is composed of one or more fact expressions. Every fact must have at least one expression.

The **column alias** stores the column name MicroStrategy uses to generate SQL statements when creating temporary tables related to the fact. Every fact must have a column alias. MicroStrategy selects a default column alias depending on the type of fact, unless you create a new column alias.

One of the example fact is 'Total Cargo Price'. It is defined on the 'CARGO' fact table's price column. It is shown in Figure 5.10

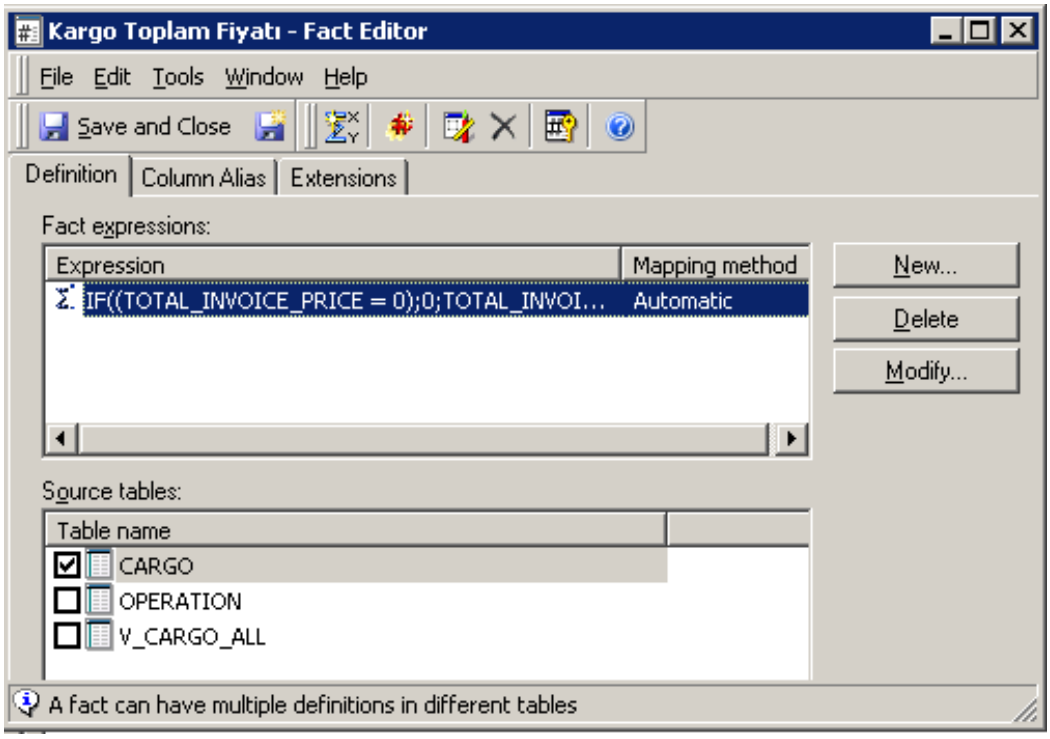

Figure 5.10 : 'Total Cargo Price' Fact

Facts of the projects are shown in Figure 5.11.

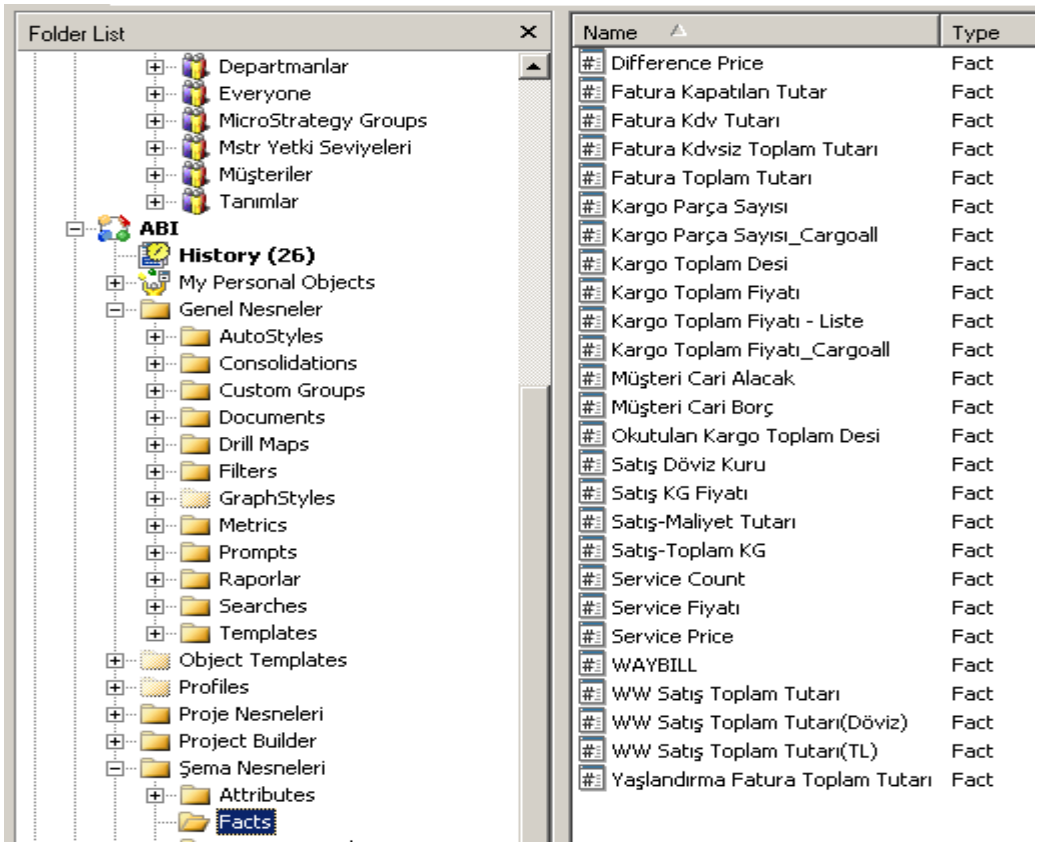

Figure 5.11 : Facts of the project

Defining Attributes:

Attributes represent the business context in which fact data is relevant. Attributes are used to define the level at which you want to view the numeric data on a report. (MicroStrategy 2010).

The business subjects have been grouped. The attributes have been created with this information. (Figure 5.12)

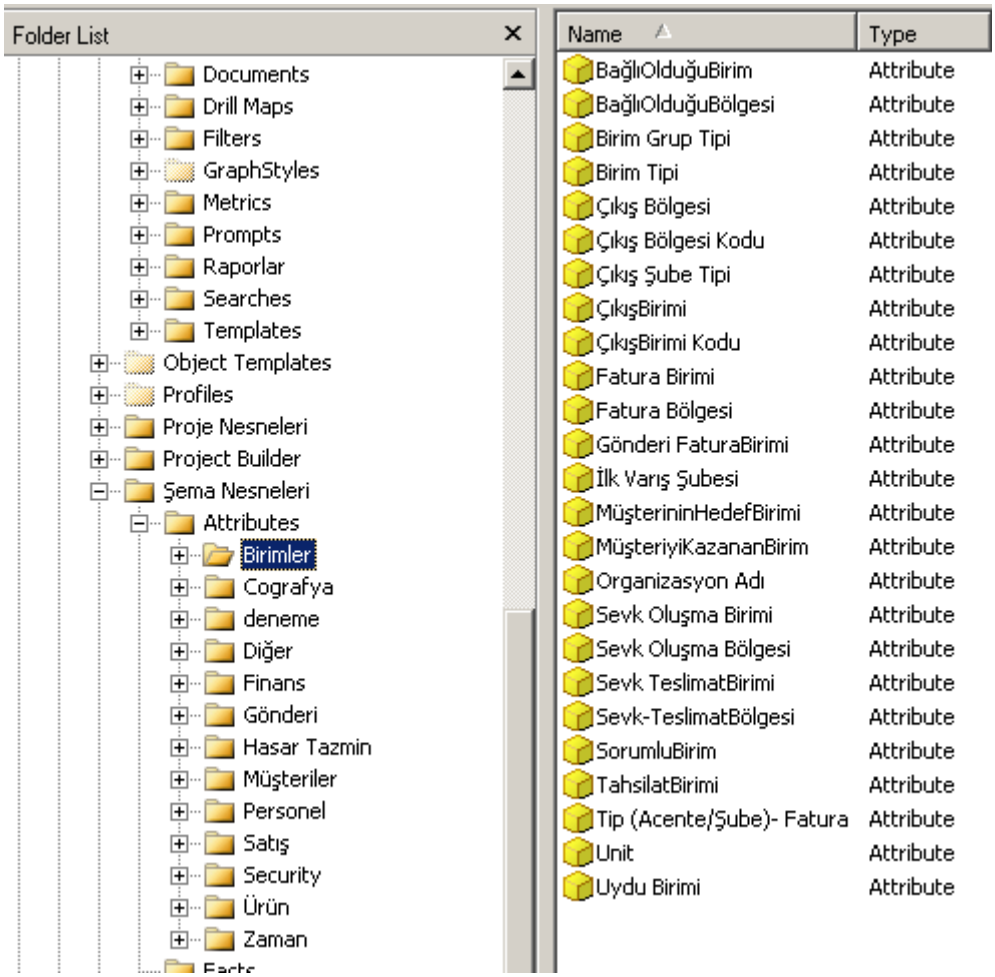

Figure 5.12 : Attributes of the project

After finishing the identifing of attributes in MicroStrategy, I determined the relation between the attributes. Attribute relationships, which are associations between attributes that specify how attributes are connected, are essential to the logical data model. Without relationships, there is no interaction between data, and therefore no logical structure. The relationships give meaning to the data by providing logical associations of attributes based on business rules.

Every direct relationship between attributes has two parts—a parent and a child. A child must always have a parent and a parent can have multiple children. The parent attribute is at a higher logical level than the child is.

In the project, between the ‘Region’ and the ‘Branch’ attributes, I defined the ‘Branch’ attribute as the child attribute of the ‘Region’ attribute. The relation between ‘Region’ and ‘Branch’ attributes can be seen in Figure 5.13

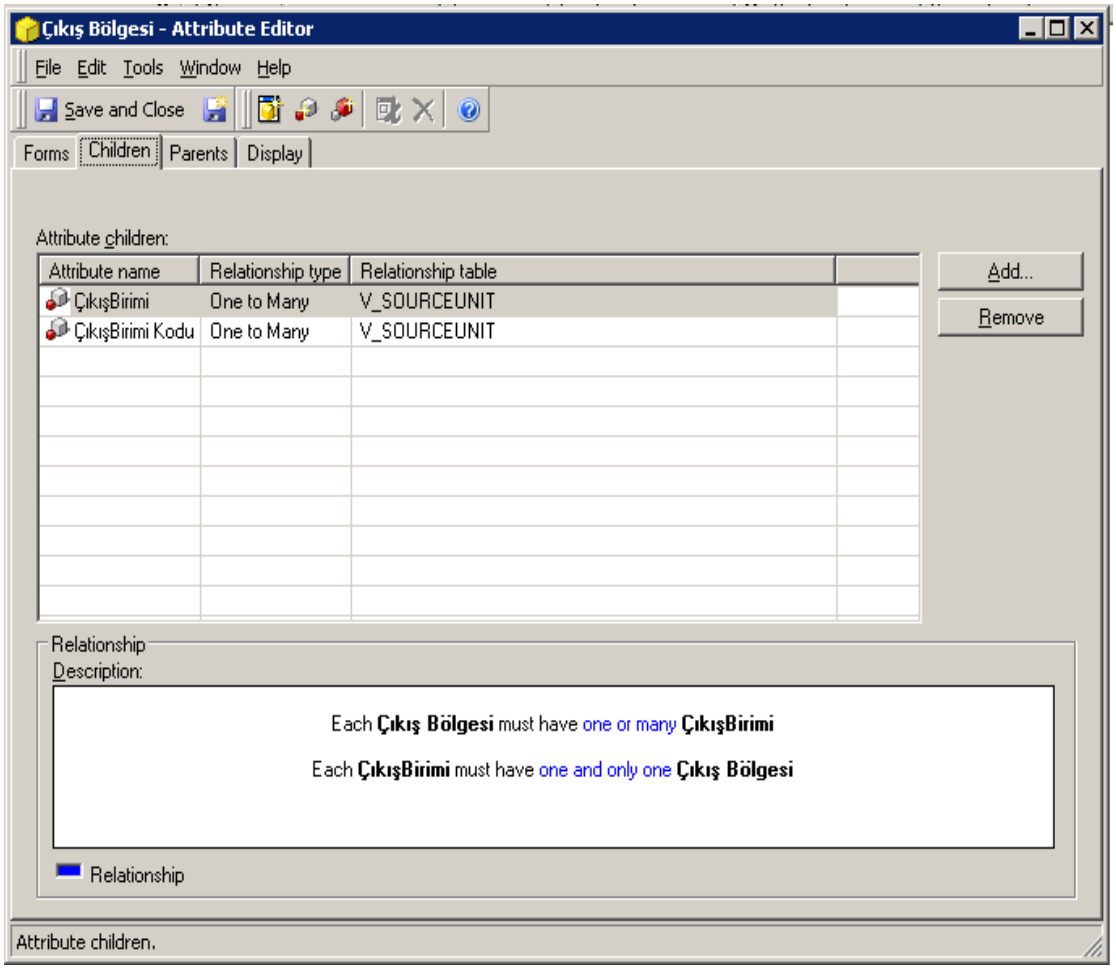

Figure 5.13 : Relation between ‘Region’ and ‘Branch’ attributes

In the data warehouse, attributes are normally identified by a unique ID column in a lookup table. Attribute relationships, which are associations between attributes that specify how attributes are connected, are essential to the logical data model. Without relationships, there is no interaction between data, and therefore no logical structure. The relationships give meaning to the data by providing logical associations of attributes based on business rules.

Every direct relationship between attributes has two parts—a parent and a child. A child must always have a parent and a parent can have multiple children. The parent attribute is at a higher logical level than the child is. (MicroStrategy 2010).

According to the warehouse model of the project, there are fact tables which were built with connecting of the relational tables. For all fact tables' unique ID column, I created an attribute. I used these attributes as a basic connection point. I connected these basic attributes between each other according to the business rules.

As I modeled the data warehouse as a hybrid structure, these connections are the relational connections between fact tables.

Relations between 'waybill', 'shipment' and 'fatura' (represents Invoice data) attributes are shown in the Figure 5.14.

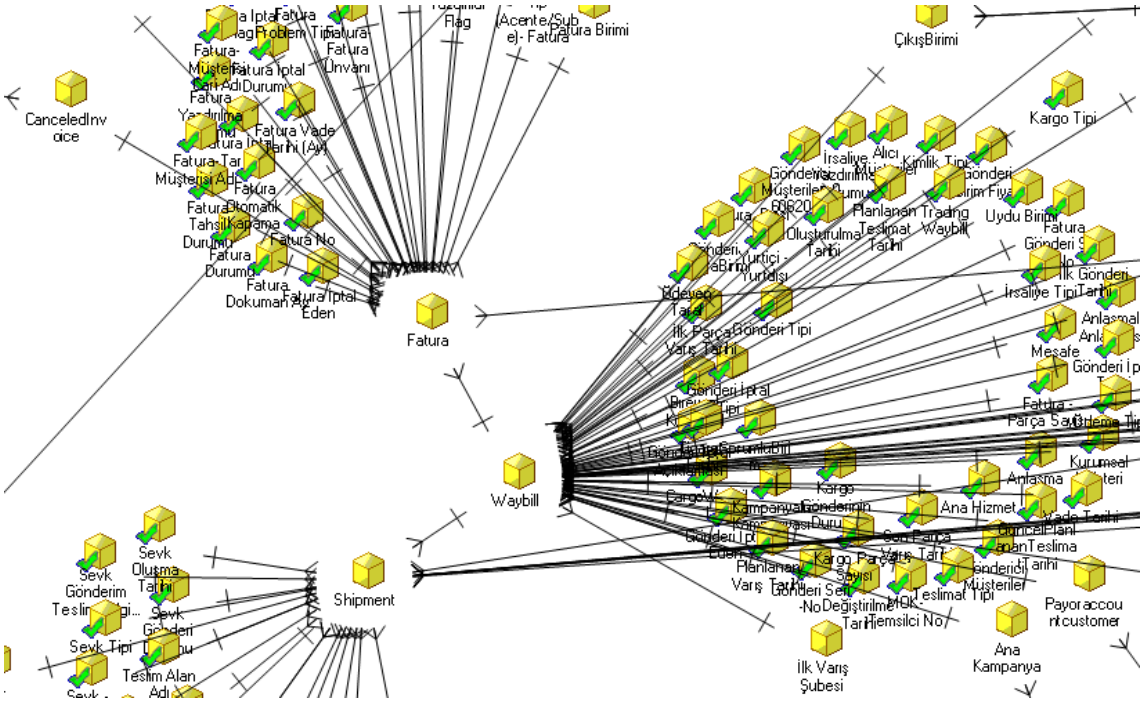

Figure 5.14 : Relations between 'waybill', 'shipment' and 'fatura' attribute

In the Figure 5.15, these relationships are seen between the attributes:

- 'Waybill' attribute (represents cargo data) may have one or more Shipment and one shipment can have only one cargo data. In this relation, 'Shipment' attribute's the child of the 'Waybill' attribute.
- 'Fatura' attribute (represents invoice data) may have one or more Waybill and one waybill can have only one Fatura data. In this relation, 'Waybill' attribute's the child of the 'Fatura' attribute.

After I connected the fact table's attributes to each other, then I connected these fact table's attributes to the dimension/lookup table's column's attributes separately with creating parent-child relationship between the attributes.

In the data warehouse as a hybrid structure, these connections are the star shema connections.

Waybill attribute's relations with attributes are shown in the Figure 5.15.

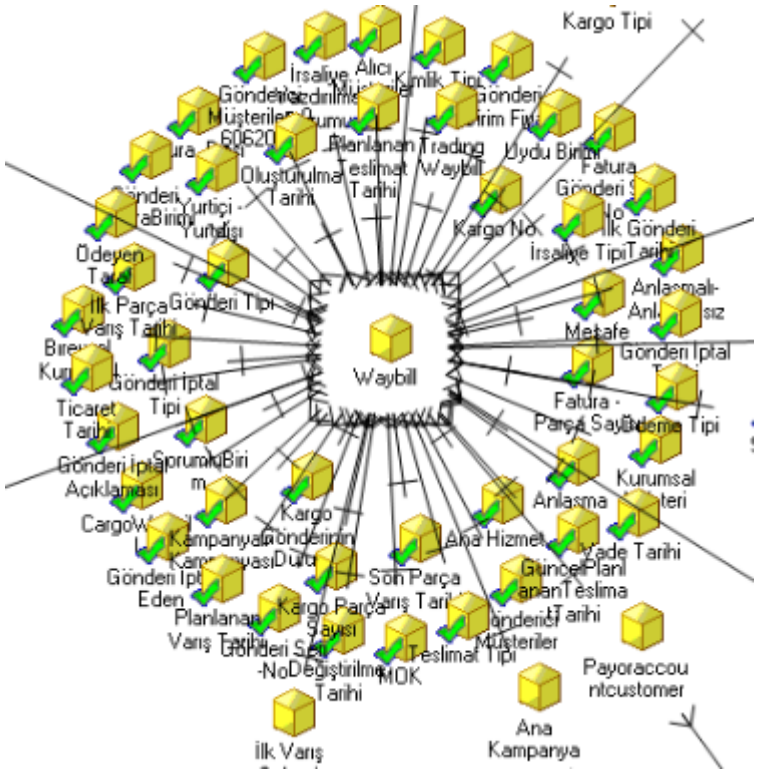

Figure 5.15 : 'Waybill' attribute relations between lookup tables.

In the Figure 5.15, it's seen that; all lookup tables's attributes which are connected to 'Waybill' attribute, may have more than one 'waybill' data but 'waybill' attribute may have only one data of the lookup table's attributes. This shows, 'Waybill' attribute is the child attributes of all lookup table's which are connected to it.

In a example of regional sales in the Istanbul region, Istanbul region represents the attribute or context of the sales unit data. It's shown in the Figure 5.16.

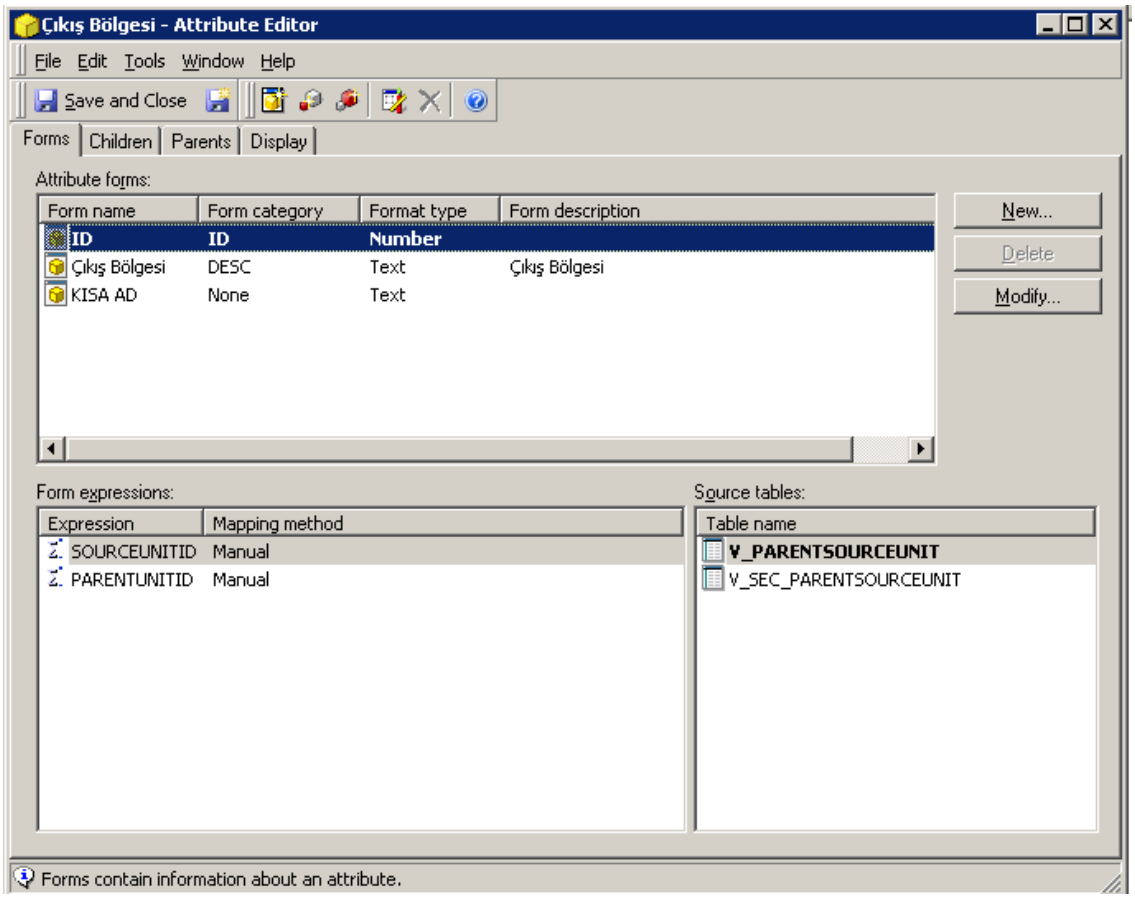

Figure 5.16 : Defining the region attribute

The region attribute's relationship with the branch attribute which is connected to the waybill attribute (that represents the cargo data) is shown in the Figure 5.17. (The region attribute is named as 'Cikis Bolgesi, branch attribute is named as 'Cikis Subesi'.)

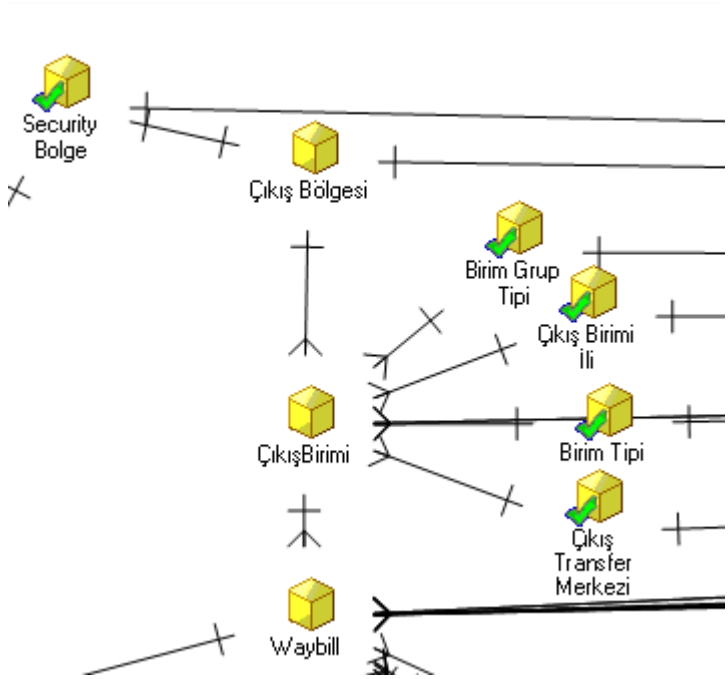

Figure 5.17 : Region, branch and cargo attributes

In the Figure 5.17, these relationships are seen between the attributes:

- Waybill attribute (represents cargo data) has only one branch (‘CikisBirimi’) and one branch can have more than one cargo data. In this relation, ‘Waybill’ attribute’s the child of the ‘CikisBirimi’ attribute.
- The Branch (‘CikisBirimi’) attribute has got also one-to-many connections to ‘BirimGrupTipi’, ‘CikisBirimiIli’, ‘BirimTipi’, ‘CikisTransferMerkezi’ attributes. ‘CikisBirimi’ attribute is the child attribute of the ‘BirimGrupTipi’, ‘CikisBirimiIli’, ‘BirimTipi’, and ‘CikisTransferMerkezi’ attributes.

As seen in Figure 5.17, there’s a hierarcy between the attributes, ‘waybill’, ‘CikisBirimi’ and ‘CikisBölgesi’.

In the data warehouse as a hybrid structure, these connections are the snowflake shema connections.

When I defined all relations between the attributes, object’s relations also become the hybrid model. It’s shown in the Figure 5.18.

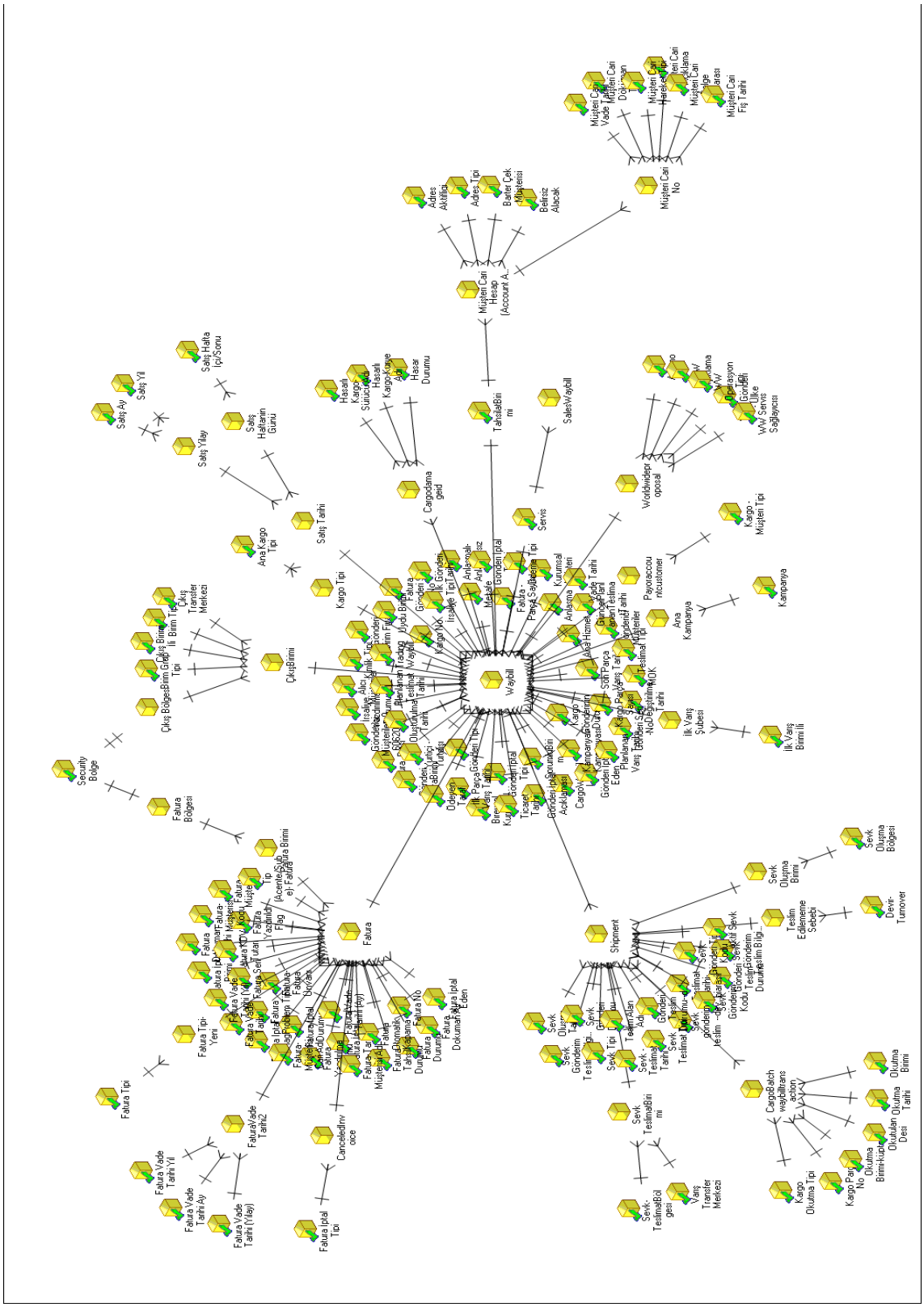

Figure 5.18 : Attribute's relations

Defining Metrics:

Metrics are objects that represent business measures and key performance indicators. They are the calculations performed on data stored in your database, the results of which are displayed on a report. Metrics are similar to formulas in spreadsheet software. (MicroStrategy 2010).

All metrics require the Mathematical formula which determines the data to be used from our data source and the calculations to be performed on that data.

An example of the formula of a metric is: Sum (Price)

When more than one attribute is on a report, as is generally the case, a metric is calculated by default at the level of the lowest-level attribute that is on the report. The lowest level is usually the attribute that reflects the least-inclusive business concept.

Metrics are defined on the fact objects.

As an example, for using summary of the total price of the cargos in the report, I created a metric object which name is 'Gonderi Toplam Tutarı'. Figure 5.19 shows the 'Gonderi Toplam Tutarı' metric.

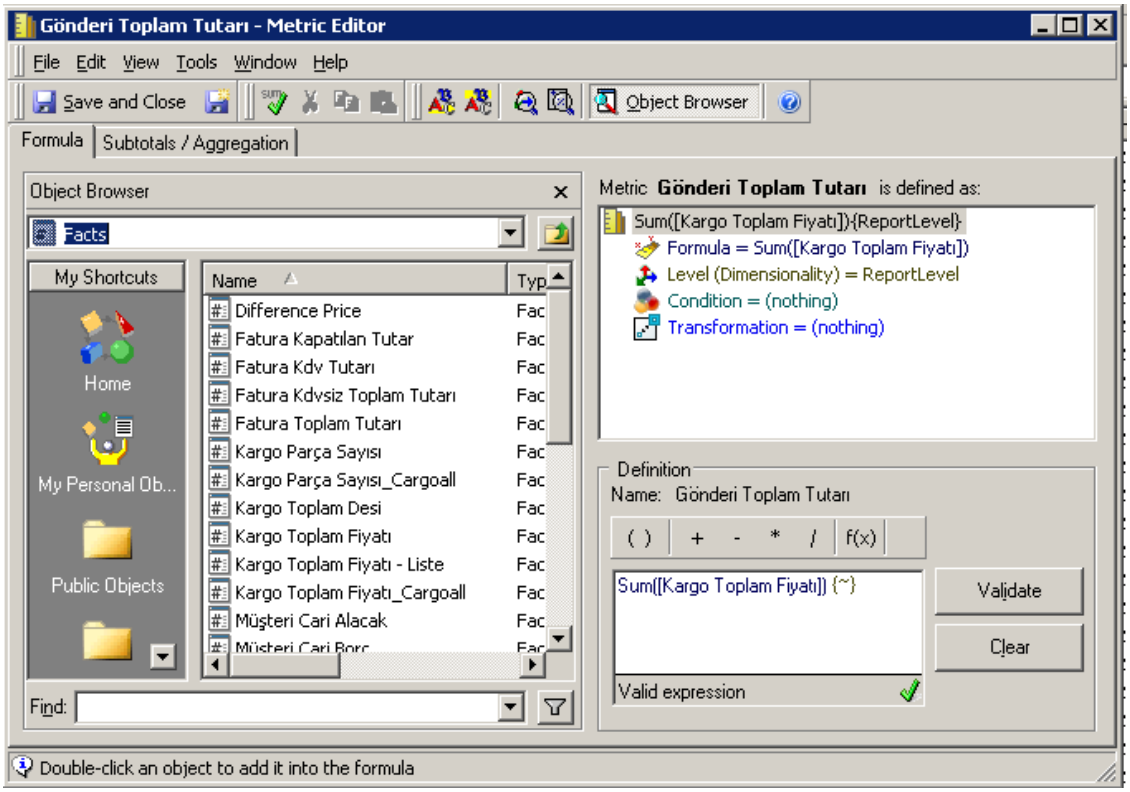

Figure 5.19 : 'Gonderi Toplam Tutarı' metric

5.4.3 Creating OLAP Cubes

Intelligent Cubes are multi-dimensional cubes (sets of data) that allow us to use OLAP Services features on reports, as well as share sets of data among multiple reports. (MicroStrategy 9 2010).

With Intelligent Cubes, a specific set of data is returned from the data warehouse. Users can create reports that display and analyze a subset of the set of data defined in an Intelligent Cube as shown Figure 5.20.

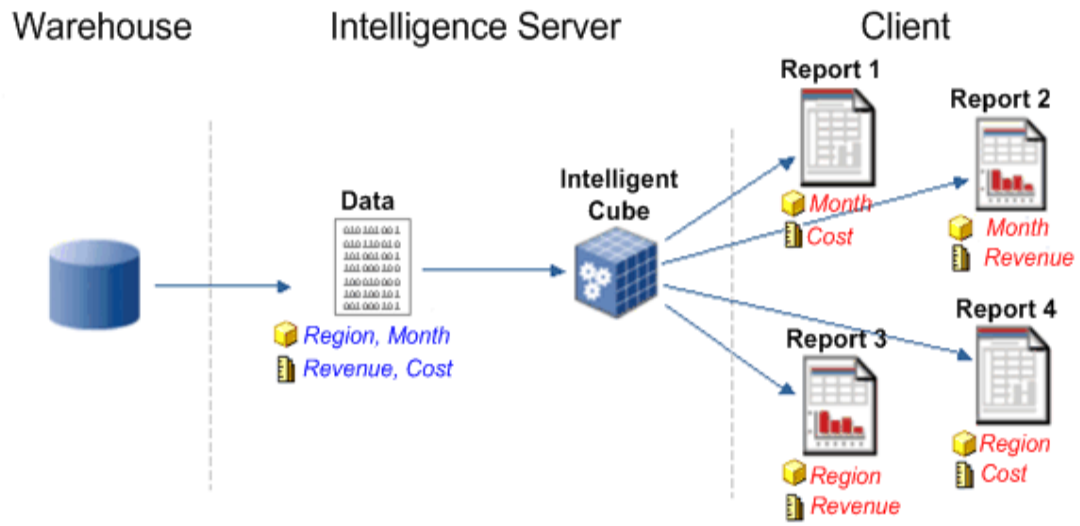

Figure 5.20 : Intelligent cube usage

Reports that connect to an Intelligent Cube can perform reporting and analysis manipulations within the Intelligent Cube without hitting the data warehouse. These manipulations are executed much faster than running a new query against a data warehouse.

Before defining the Intelligent Cube, these questions must be answered first:

- What subset of business queries does the Intelligent Cube need to provide data for? Intelligent Cubes allows to create sets of data that can support multiple reports that answer variations to similar business queries.
- Are there any reports that currently access the data warehouse that could benefit from accessing an Intelligent Cube instead?

For creating intelligent cubes in the project, first all departments prepared 210 report request forms. With these forms, I defined all users' needs. I grouped their requests into the functional sentences. With this information, cube contents occurred as cargo cube, sales cube, shipment cube, cargo transaction cube etc.

Before creating cubes, the attribute objects and metric objects were prepared. According to the need of data which will be used in the cube and also according to the server's capability, cube filters have been defined.

With using the Microstrategy tool, I choosed the objects and defined the cubes.

The Cargo Cube is shown in Figure 5.21.

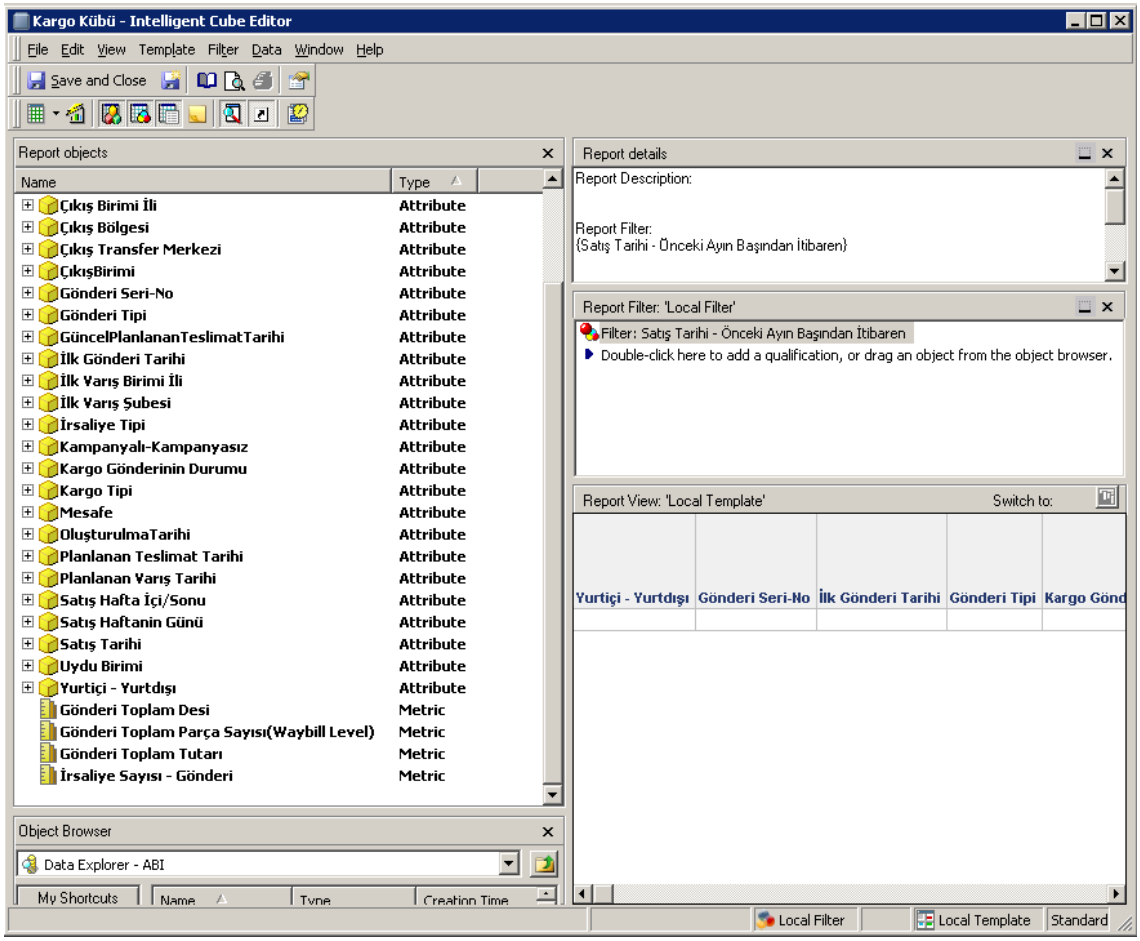

Figure 5.21 : Cargo Cube

When I selected the attributes and metrics and run the cube, DSS tool creates a SQL query and run it.

Figure 5.22 shows the Cargo Cube SQL.

```
select   distinct pa13.WORLDWIDE WORLDWIDE, a122.NAME NAME, pa13.SERIAL_NUMBER SERIAL_NUMBER,
         pa13.SHIPMENTDATEID SHIPMENTDATEID,a123.shipmentdate shipmentdate, pa13.LOVSHIPMENTTYPEID
         LOVSHIPMENTTYPEID, a112.NAME NAME0,pa13.LOVSHIPMENTSTATUSID LOVSHIPMENTSTATUSID, a115.NAME
         NAME1, pa13.LOVWAYBILLTYPEID LOVWAYBILLTYPEID, a124.NAME NAME2, pa13.LOVPACKTYPEID LOVPACKTYPEID,
         a116.NAME NAME3, a16.PARENTUNITID PARENTSOURCEUNITID,SUBSTRING(a126.NAME, 1, (LEN(a126.NAME) - 6)) CustCol_6,
         pa13.SOURCEUNITID SOURCEUNITID, a16.NAME NAME5, pa13.LOVUNITDISTANCETYPEID LOVUNITDISTANCETYPEID,
         a117.NAME NAME6, pa13.contract_f contract_f, a111.NAME NAME7, pa13.campaign_f campaign_f,
         a114.NAME NAME8, pa13.SERVICEID SERVICEID, a18.NAME NAME9, pa13.ACTUALPLANNEDDELIVERYDATEID
         ACTUALPLANNEDDELIVERYDATEID,a113.actualplanneddeliverydate actualplanneddeliverydate, pa13.AUDITCREATEDATEID
         AUDITCREATEDATEID,a118.auditcreatedate auditcreatedate, pa13.PLANNEDDELIVERYDATEID PLANNEDDELIVERYDATEID,
         a119.planneddeliverydate planneddeliverydate,pa13.PLANNEDARRIVALDATEID PLANNEDARRIVALDATEID,
         a120.plannedarrivaldate plannedarrivaldate,pa13.OPERATIONDATEID OPERATIONDATEID,
         a14.OPERATION_DATE OPERATION_DATE,pa13.LOVPACKTYPEID0 LOVPACKTYPEID0, a110.NAME NAME10,
         a14.DAYOFWEEKID DAYOFWEEKID,a15.NAME NAME11,a15.WEEKDAY WEEKDAY,a15.WEEKDAY WEEKDAY0,
         pa13.BIREYSEL_KURUMSAL BIREYSEL_KURUMSAL,pa13.campaignconditionid campaignconditionid,
         a19.campaign_name campaign_name,pa13.DELIVERYUNITID DELIVERYUNITID,a17.NAME NAME12,
         a16.SENDERHUBUNITID SENDERHUBUNITID,a127.NAME NAME13,pa13.SATELLITEID SATELLITEID,a121.NAME NAME14,
         a16.hintcityid hintcityid,a125.CITY CITY,a17.hintcityid hintcityid0,a128.CITY CITY0,pa13.WJXBFS1 WJXBFS1
from     #ZZMD00 pa13
         join   OPERATIONDATE      a14 on  (pa13.OPERATIONDATEID = a14.OPERATIONDATEID)
         join   DAYOFWEEK          a15 on  (a14.DAYOFWEEKID = a15.DAYOFWEEKID)
         join   V_SOURCEUNIT       a16 on  (pa13.SOURCEUNITID = a16.SOURCEUNITID)
         join   dbo.v_firstdeliveryunit    a17 on  (pa13.DELIVERYUNITID = a17.DELIVERYUNITID)
         join   V_PRIMARYSERVICES  a18 on  (pa13.SERVICEID = a18.PRIMARYSERVICEID)
         join   V_CAMPAIGNCONDITION a19 on  (pa13.campaignconditionid = a19.campaignconditionid)
         join   dbo.LOVMAINPACKTYPE a110 on  (pa13.LOVPACKTYPEID0 = a110.LOVMAINPACKTYPEID)
         join   V_CONTRACT_F       a111 on  (pa13.contract_f = a111.contract_f)
         join   LOVSHIPMENTTYPE    a112 on  (pa13.LOVSHIPMENTTYPEID = a112.LOVSHIPMENTTYPEID)
         join   V_ACTUALPLANNEDDELIVERYDATE a113 on (pa13.ACTUALPLANNEDDELIVERYDATEID =
a113.ACTUALPLANNEDDELIVERYDATEID)
         join   V_CAMPAIGN_F       a114 on  (pa13.campaign_f = a114.campaign_f)
         join   LOVSHIPMENTSTATUS  a115 on  (pa13.LOVSHIPMENTSTATUSID = a115.LOVSHIPMENTSTATUSID)
         join   LOVPACKTYPE        a116 on  (pa13.LOVPACKTYPEID = a116.LOVPACKTYPEID)
         join   LOVUNITDISTANCETYPE a117 on  (pa13.LOVUNITDISTANCETYPEID = a117.LOVUNITDISTANCETYPEID)
         join   V_AUDITCREATEDATE  a118 on  (pa13.AUDITCREATEDATEID = a118.AUDITCREATEDATEID)
         join   V_PLANNEDDELIVERYDATE a119 on  (pa13.PLANNEDDELIVERYDATEID = a119.PLANNEDDELIVERYDATEID)
         join   V_PLANNEDARRIVALDATE a120 on  (pa13.PLANNEDARRIVALDATEID = a120.PLANNEDARRIVALDATEID)
         join   dbo.V_SATELLITE    a121 on  (pa13.SATELLITEID = a121.SATELLITEID)
         join   V_WORLDWIDE        a122 on  (pa13.WORLDWIDE = a122.WORLDWIDE)
         join   V_SHIPMENTDATE     a123 on  (pa13.SHIPMENTDATEID = a123.SHIPMENTDATEID)
         join   LOVWAYBILLTYPE     a124 on  (pa13.LOVWAYBILLTYPEID = a124.LOVWAYBILLTYPEID)
         join   dbo.v_sourceunitcity a125 on  (a16.hintcityid = a125.hintcityid)
         join   V_PARENTSOURCEUNIT a126 on (a16.PARENTUNITID = a126.SOURCEUNITID
         join   dbo.V_SENDERHUBUNIT a127 on  (a16.SENDERHUBUNITID = a127.SENDERHUBID)
         join   dbo.v_firstdeliveryunitcity a128 on  (a17.hintcityid = a128.hintcityid)
```

Figure 5.22 : Cargo Cube SQL

When the cube is created, DSS server keeps the data on the memory until the cube is run again. Until that time, the data of the cube can be used in reports for analyzing which is called from memory. With this technique, users make their analyze very fast.

5.4.4 Creating Reports

Reports display the business data, and are the focus of decision support system. We perform data analysis on reports to gather business insight. The results displayed in any DSS report are often a starting point for further investigation. (MicroStrategy 9 2010).

In reports, I used attributes and metric objects to show the data. I prepared some reports with using directly data warehouse's objects, some of them was prepared with using cube objects.

The Figure 5.23 shows the design of an example report which runs the giro data of the branches. In the report design, used attributes, metrics and filters can be seen.

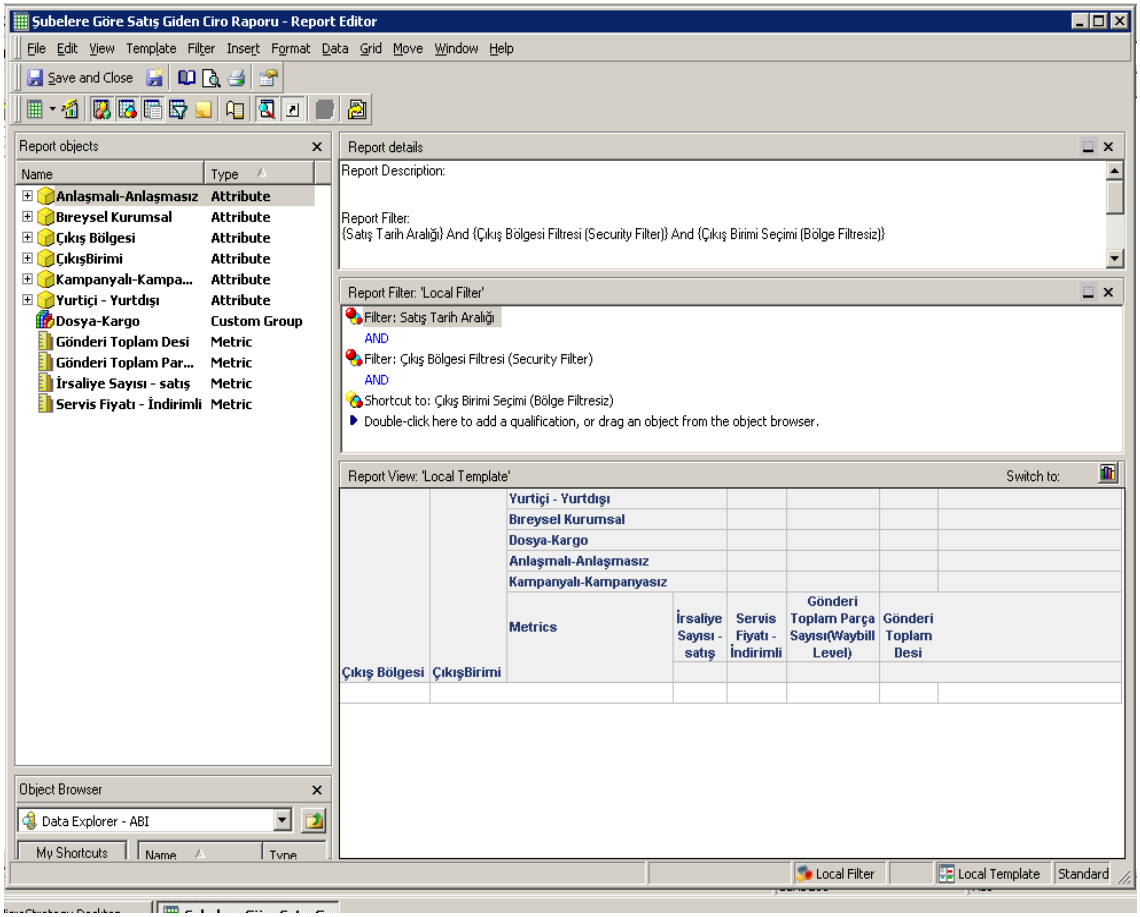

Figure 5.23: Design of the 'Branch's giro report'

When the report is started, the SQL is run by the DSS server which is seen in Figure 5.24.

```
select  a11.SOURCEUNITID  SOURCEUNITID,a11.WORLDWIDE  WORLDWIDE,a11.campaign_f  campaign_f,
        a11.BIREYSEL_KURUMSAL  BIREYSEL_KURUMSAL,  a11.contract_f  contract_f,
        count(distinct (Case when a11.LOVPACKTYPEID  in (1) then a11.WAYBILLID  else NULL end))  WJXBFS1,
        sum((Case when a11.LOVPACKTYPEID  in (1) then a11.TOTAL_VOLUME  else NULL end))  WJXBFS2,
        sum((Case when a11.LOVPACKTYPEID  in (1) then a11.PIECE_COUNT  else NULL end))  WJXBFS3,
        max((Case when a11.LOVPACKTYPEID  in (1) then 1 else 0 end))  GODWFLAG1_1,
        count(distinct (Case when a11.LOVPACKTYPEID  not in (1) then a11.WAYBILLID  else NULL end))  WJXBFS4,
        sum((Case when a11.LOVPACKTYPEID  not in (1) then a11.TOTAL_VOLUME  else NULL end))  WJXBFS5,
        sum((Case when a11.LOVPACKTYPEID  not in (1) then a11.PIECE_COUNT  else NULL end))  WJXBFS6,
        max((Case when a11.LOVPACKTYPEID  not in (1) then 1 else 0 end))  GODWFLAG4_1
into #ZZMD00
from    dbo.CARGO       a11
        join    INVOICE a12  on  (a11.INVOICEID  = a12.INVOICEID)
        join    V_I_INVOICEUNIT         a13  on  (a12.unitid = a13.i_invoiceunitid)
        join    V_SOURCEUNIT  a14  on  (a11.SOURCEUNITID = a14.SOURCEUNITID
        join    V_SEC_PARENTSOURCEUNIT      a15  on  (a14.PARENTUNITID  = a15.SOURCEUNITID)
        join    OPERATIONDATE           a16  on  (a11.OPERATIONDATEID  = a16.OPERATIONDATEID)
where   (a16.OPERATION_DATE  between '2011-08-01' and '2011-08-22'
and (a14.PARENTUNITID  = a15.SOURCEUNITID
or a13.PARENTUNITID  = a15.SOURCEUNITID)
and a11.SOURCEUNITID  in (782, 627, 414)
and (a11.LOVPACKTYPEID  in (1)
or a11.LOVPACKTYPEID  not in (1)))
group by a11.SOURCEUNITID,a11.WORLDWIDE,a11.campaign_f,a11.BIREYSEL_KURUMSAL,a11.contract_f

select  distinct a13.PARENTUNITID  PARENTSOURCEUNITID,a17.NAME  NAME,pa12.SOURCEUNITID
        SOURCEUNITID,a13.NAME  NAME0,pa12.WORLDWIDE  WORLDWIDE,a16.NAME  NAME1,
        pa12.BIREYSEL_KURUMSAL  BIREYSEL_KURUMSAL,pa12.contract_f  contract_f,a14.NAME  NAME2,
        pa12.campaign_f  campaign_f,a15.NAME  NAME3,pa12.WJXBFS1  WJXBFS1,
        pa12.WJXBFS3  WJXBFS2,pa12.WJXBFS2  WJXBFS3
from    #ZZMD00         pa12
        join    V_SOURCEUNIT  a13  on  (pa12.SOURCEUNITID  = a13.SOURCEUNITID)
        join    V_CONTRACT_F  a14  on  (pa12.contract_f = a14.contract_f)
        join    V_CAMPAIGN_F  a15  on  (pa12.campaign_f = a15.campaign_f)
        join    V_WORLDWIDE  a16  on  (pa12.WORLDWIDE = a16.WORLDWIDE)
        join    V_PARENTSOURCEUNIT  a17  on  (a13.PARENTUNITID  = a17.SOURCEUNITID  and
        a13.lovorganizationid = a17.lovorganizationid)
where   pa12.GODWFLAG1_1 = 1

select  distinct a13.PARENTUNITID  PARENTSOURCEUNITID,a17.NAME  NAME,
        pa12.SOURCEUNITID  SOURCEUNITID,a13.NAME  NAME0,pa12.WORLDWIDE  WORLDWIDE,
        a16.NAME  NAME1,pa12.BIREYSEL_KURUMSAL  BIREYSEL_KURUMSAL,pa12.contract_f  contract_f,
        a14.NAME  NAME2,pa12.campaign_f  campaign_f,a15.NAME  NAME3,pa12.WJXBFS4  WJXBFS1,
        pa12.WJXBFS6  WJXBFS2,pa12.WJXBFS5  WJXBFS3
from    #ZZMD00         pa12
        join    V_SOURCEUNIT  a13  on  (pa12.SOURCEUNITID  = a13.SOURCEUNITID)
        join    V_CONTRACT_F  a14  on  (pa12.contract_f = a14.contract_f)
        join    V_CAMPAIGN_F  a15  on  (pa12.campaign_f = a15.campaign_f)
        join    V_WORLDWIDE  a16  on  (pa12.WORLDWIDE = a16.WORLDWIDE)
        join    V_PARENTSOURCEUNIT  a17  on  (a13.PARENTUNITID  = a17.SOURCEUNITID  and
        a13.lovorganizationid = a17.lovorganizationid)
where   pa12.GODWFLAG4_1 = 1
```

Figure 5.24 : 'Branch's giro report's SQL

Reporting on an Intelligent Cube provides quick access to data, as the data has been pre-aggregated. This returns results much faster than querying the data warehouse. Reporting on Intelligent Cubes also allows you to use all of the OLAP Services features,

including derived elements, which allow you to group attribute elements in a report on the fly, to provide a new view of report data for analysis and formatting.

5.4.5 Creating Dashboards

A dashboard is a kind of document, commonly one page long and usually viewed online. Dashboards contain interactive features that allow analysts to control how they view data. Each user can interact with the dashboard to display only the data they are interested in (using panels and selectors) or only specific attribute elements or metrics (using a selector). (MicroStrategy 9 2010).

Dashboards are often used to assess performance, to provide a quick status check, or to monitor contributions to overall goals of the business. Dashboards summarize key business indicators by presenting them in visually intuitive, easy-to-read, interactive documents.

An example of a dashboard which created for the giro analysis using distances information is seen on Figure 5.25

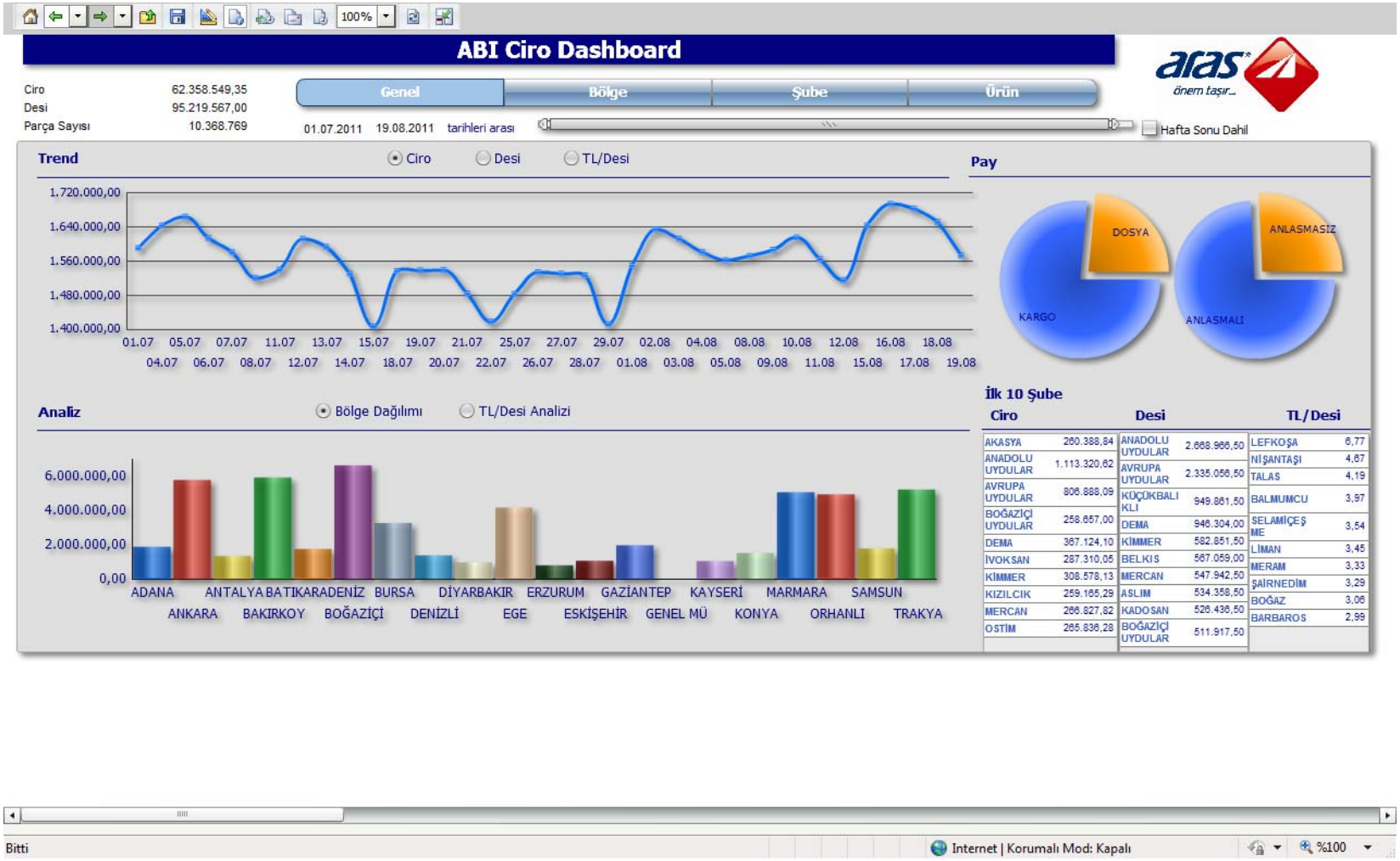

Figure 5.25 : Giro analysis dashboard

5.5 VERIFYING DATA CONSISTENCY

One of the most important subject of developing a system that uses data is the verifying it's data consistency. The data of the DSS system's reports is compared with the source OLTP system's reports.

It's seen that the data in the DSS is consistent with the OLTP system. There's no data loss or data derivation in the ETL processes and the analyzing layer. It's seen in Table 5.3.

Report Name	OLTP System	DSS
Total giro of August, 2011	70563185	70563185
Total cargo count of August, 2011	9033379	9033379
Total cargo piece count of August, 2011	11638905	11638905

Table 5.3 : Comparison of the data between DSS and OLTP System

5.6 BENEFITS OF THE NEW SYSTEM

I analyzed the new queries' development time in the DSS and the performance of the running queries. I measured the time of the processes in DSS and also in the OLTP System. I verified a big performance of the DSS system according to the OLTP system.

Besides the performance, it's also seen that there are many capabilities, they can be done in the DSS system which can't be provided by the OLTP system.

5.6.1 Verification of the Developing Time

When I compared the developing time of a report between the OLTP system and the DSS system, it's seen that there's a dramatic differences between them.

In the OLTP system, one report can be built in 4-5 hours by a developer, on the avarage.

For building a report in OLTP, these steps are performed:

- First, SQL string is created in a procedure packet on the database server.
- In the report builder, report is designed.
  When designing a report in the OLTP system, all columns are described with referencing to the dealing database table's columns. Sometimes, a column definition is not enough for using as a report column. In the report, there may also be need to create some functions or some consolidation of more than one table columns. Every time, these definitions are created for building a new report.
- The new report which has been designed in the report builder is deployed and replaced in the report server.
- The new report's menu option is created in the automation system which is Windows application modeled.
- The automation system is built and published to the client's machines.
- User privileges are described in the automation system's security management tool.

After these steps, users can see the new report's menu option in the automation system's menu.

For building the same report with the DSS system, a builder can build a report in a few minutes. These are the steps of creating report in DSS:

- The report builder drags and drops the attributes and the filters of the DSS which are needed, to the new blank report. (Most of the attributes were created at the beginning when the data warehouse DSS system was modeled. For using a new reports, they are just dragged and dropped to the new report).
  With this process, the SQL string and report conditions are created automatically by the DSS system.
- The new report is saved and user pfivileges are described in the DSS system.

Users can see and use the new report in the DSS system.

The comparision of developing a report between DSS and OLTP System can be seen in Table 5.4.

	OLTP System	DSS
Creating SQL string	60 minutes	-
Designing report	180 minutes	10 minutes
Deploying report	20 minutes	-
Creating menu option and building windows application structured automation system	30 minutes	-
Describing user privileges	10 minutes	10 minutes
Total development time	**300 minutes**	**20 minutes**

Table 5.4 : Development time on DSS and OLTP

5.6.2 Verification of the Query Performance

I measured the queries and reports' running time in both OLTP System and DSS System. The results showed that DSS system provides great performance according to the OLTP System. The values are seen in Table 5.5

Executed Jobs	OLTP System	DSS uses Datawarehouse Database	DSS uses OLAP Cubes
Monthly Delivery Performance Report Run	3 days	1 hour	8 seconds
Monthly Agency Progress Payment Calculations	3,5 days	1 hour 20 min.	10 seconds
Monthly Giro Report Run	5 hours	10 minutes	4 seconds

Table 5.5 : Running queries' period

5.6.3 DSS's Extra Capabilities and Advantages

Many requirements of the transportation company's departments that the OLTP system couln't have been the answer, can be done in the warehouse DSS.

With the DSS, managers can analyze one of the most important subject of the company which is the units' effect to the giro with realizing the 'distance' and 'giro/volume' parameters, which can't be done in the OLTP system. It is seen in Figure 5.26.

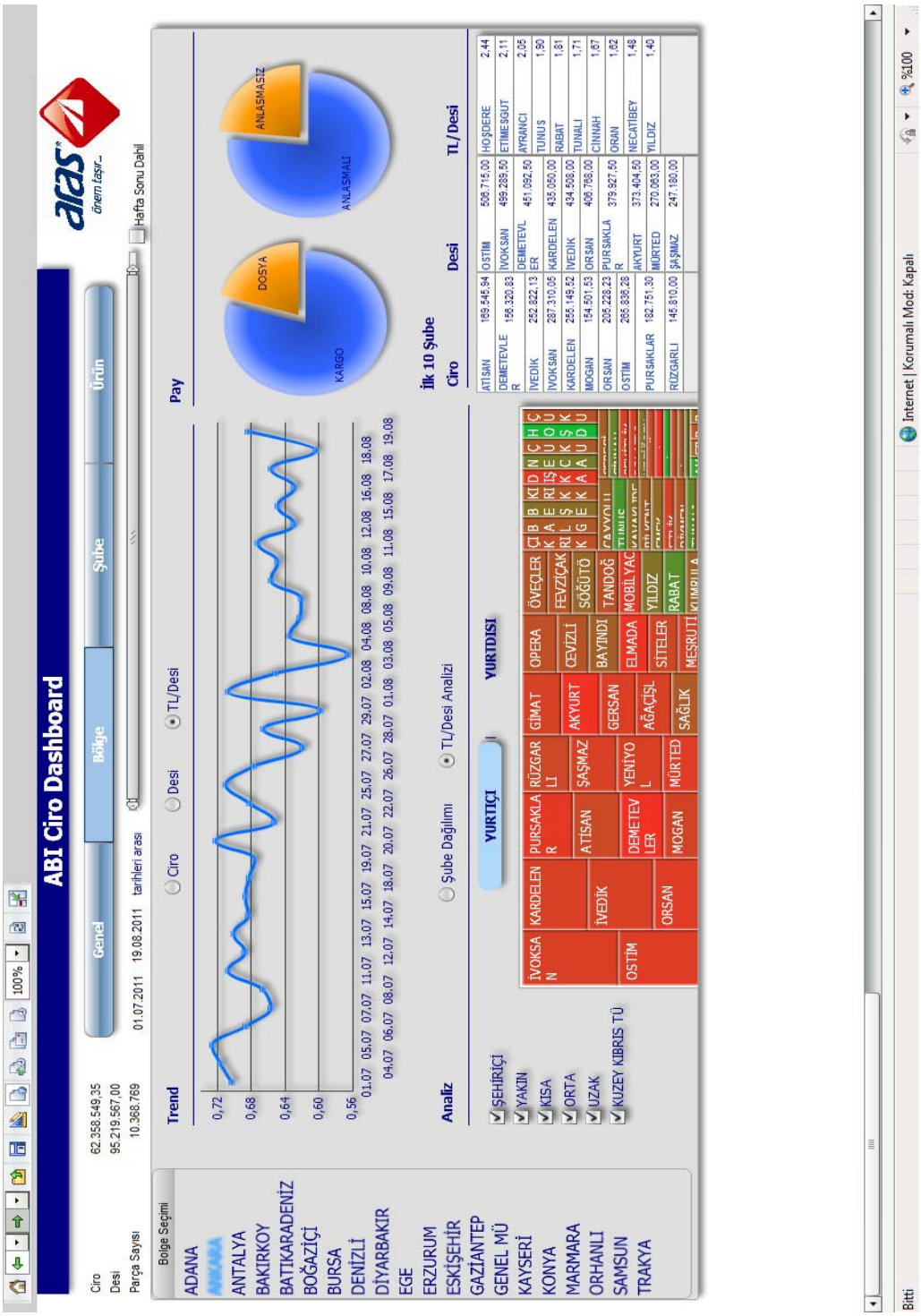

Figure 5.26 : Unit's giro effects with the 'distance' and 'giro/volume' parameters

Decision makers of the company can also take trend analysis reports for year's period of time.

Operation Department runs shipment analysis can cargo transaction analysis.

Financial department can take customer balance reports for whole firm's customer just in minutes.

Sales department can analyze the sales according to the distance, cargo type, discount rate of the customer's contract, sectors etc. With these analyses, they can decide the rate of the price's rise analytically. Sales managers also design different kinds of campaigns in different terms of the year with the ability of analyzing giro of the firm.

For special customers, user accesses are defined. Customers have begun to analyze their own shipment data with using OLAP cubes of the project in their firms with using Web interface.

With the DSS system, departments of the company have begun to analyze their business.

CONCLUSION

In the changing and developing transportation sector, organizations needs to make decision very agile and fast about the complex situations. The necessity of using big amount of data for making decisions, they need dynamic and analytical structured analysis systems. According to their operational systems and their data, the model and the technology of the DSS solution must be the answer for performenced analysis with many parameters. Using the short time's period data isn't enough for decision makers in the transportation sector anymore. Trend analysis are needed which must have multidimensional structure.

In this thesis, a transportation company was analyzed and the needs of the company were determined. The company's problems were also listed.

For being an answer to the transportation company's decision maker's problem, it's decided that the most suitable model is the Hybrid OLAP data warehouse model.

After modeling the warehouse structure, the analyzing tool is selected. With the analyzing tool, business intelligence layer is structured.

With verifying the warehouse structured DSS, the company succeded to make it's analysis very efficiently and agile. Decision makers of the firm can analyze the data they need.

Finance department is planning to develop 'the activity based costing' analyzsis.

Operation department will begin to work on route optimization for the company next year. Operation department will also use the DSS' geographical data for defining new units' and hubs' destinations.

Sales department has begun to develop the functional spects about a CRM project which will use the DSS analysis.

REFERENCES

Books:

Englewood Cliffs, N.J., (1997) Prentice-Hall. ISBN 0-130-86215-0

Hoffer, Jeffrey A. (2007) Modern Systems Analysis and Design, 5th edition, Prentice Hall

Inmon, William H. (2005) Building the Data Warehouse , Fourth Edition, Wiley Publishing ISBN: 978-0-7645-9944-6

Inmon, William H. (2002) Building the Data Warehouse , John Wiley & Sons Publishers.

Keen, Scott-Morton (1978). Decision Support System.

Khan, Arshad (2003). Data Warehousing 101 Concepts and Implementation, IUniverse, ISBN: 0-595-29069-8

Loshin, D. (2003). Business Intelligence: The Savvy Manager's Guide. Morgan Kaufmann Publishers.

Microsoft MSDN (2005), Online Transaction Processing vs. Decision Support

MicroStrategy 9. (2010). Project Design Guide, Version: 9.0.2. Document Number: 09330902

Ponniah, Paulraj. (2010). Data Warehouseing Fundamentals-Second Edition, A Comprehensive Guide for IT Professionals. ISBN: 0-471-22162-7

Power, Daniel J. (2002). Decision Support Systems: Concepts and Resources for Managers. ISBN 1-56720-497-X (alk.paper)

Power, Daniel J. (November 2, 2009). Decision Support Basics. Business Expert Press. ISBN: 1-60649-082-6

Sprague Ralph H. ,Carlson Eric D. (1982). Building effective decision support systems. Prentice-Hall

Tupper, Charless D. (2009). Data Architecture From Zen to Reality. Elsevier Inc. ISBN: 978-0-12-385126-0

Turban, Efraim, Sharda, Ramesh, Delen Dursun (2007) Decision support and BI Systems ISBN: 978-0-13-610729-3

V. Poe, P. Klauer, S. Brobst, (1998). Building A Data warehouse for Decision Support, Prentice Hall, Upper Saddle River

Other Publications

Oracle® (2007) Database 2 Day + Data Warehousing Guide 11g Release 2 (11.2)
http://download.oracle.com/docs/cd/E11882_01/server.112/e10578/tdpdw_intro.htm

Power, Daniel J. (2004). Decision Support Systems: Frequently Asked Questions
http://dssresources.com/history/dsshistoryv28.html

Why Are Operational Systems Not Suitable For DSS (2008)
http://www.datawarehousesolution.net

APPENDICES

APPENDIX A

PL/SQL Source code of ETL–Extracting Phase Procedure of Cargo Fact

```
CREATE OR REPLACE PROCEDURE PROC_wh_cargo
IS
/******************************************************************************
   NAME:       PROC_wh_cargo
   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------
   1.0        29.07.2011  Mert SUN         1. Created this procedure.

      Object Name:     PROC_wh_cargo
      Date and Time:   29.07.2011
      Username:         (set in TOAD Options, Procedure Editor)

      Description:      Extracting CARGO_FACT table

******************************************************************************/


P_DATE date;
p_transfer_date date;
p_last_transfer_date date;
p_whtransferid RAW (16);

p_countchk integer:=0;
p_operationdateid    VARCHAR2 (50);
ms_prc INTEGER;
l_rowcount integer :=0;


BEGIN

--   execute immediate 'alter session ENABLE PARALLEL DML';


    select sys_guid() into p_whtransferid from dual;
    commit;

    select

    max(whtransfer_date)-1/(60*60*24)
    into p_last_transfer_date from whtransfer w
    where w.LOVWHTRANSFERTYPEID=1  and W.LOVWHTRANSFERSTATUSID=2;

    commit;

    p_transfer_date:=sysdate;
```

```
    insert into whtransfer(whtransferid, whtransfer_date,
lovwhtransferstatusid, lovwhtransfertypeid, transfer_complete_date,
row_count)
    select p_whtransferid whtransferid, p_transfer_date
whtransfer Date, 1 lovwhtransferstatusid, 1 lovwhtransfertypeid, null
TRANSFER_COMPLETE_DATE,l_rowcount from dual;

    commit;


    PROC_SEND_MAIL('mertsun@araskargo.com.tr','warehouse-cargo: ' ||
p_transfer_date,'warehousea cargo data aktarimi basladi');


        execute_immediate('truncate table cargo_fact_tmp');


        insert into cargo_fact_tmp
        (
           cargono, operation_date, sourceunitid, serial_number,
           shipment_code, senderaccountid, receiveraccountid,
           waybillid, shipmentid, cargoid, campaignconditionid,
           accountcontractversionid, yi yd, piece count, total_volume,
           bireysel_kurumsal, lovpayortypeid, lovpacktypeid,
           primaryserviceid, total_price, total_invoice_price,
           operationdateid, satelliteid, lovshipmenttypeid,
           planned_delivery_date, actual_planned_delivery_date,
           planned arrival date, first piece arrival date,
           last_piece_arrival_date, lovunitdistancetypeid,
           duration, mobile, manifestunitid, description,
           lovshipmentstatusid, integration_code,
           contents_description, delivery_date, senderaccountname,
           senderaccountaddressname, receiveraccountname,
           receiveraccountaddressname, deliveryunitid,
      lovdeliverytypeid,
           responsibleunitid, shipment_date, shipmentdeliveredid,
           lovdeliveryfailurereasonid, parentdeliveryunitid,
           parentsourceunitid, cancel_date, canceledby, canceled,
           audit_create_date, auditcreateunitid, audit_modify_date,
           audit_deleted, invoiceid, cancelunitid, lovwaybilltypeid,
           cancel_description, auditcreatedby, auditmodifiedby,
           auditmodifyunitid, senderaccountaddressversionid,
           invoiceunitid, collectionunitid, invoiceaddressversionid,
           worldwide, lovdocumentprintstatusid, diffinvoiceid,
           payoraccountcustomerid, returned, puantumbonus,
           puantumcancel, puantumcanceled, puantumcardno,
           puantumcardowner, puantumweb, due_date, payoraccountname,
           invoiceaddressid, acccontractid, receiveraddressversionid,
           refcode, trading waybill number, trading goods,
           responsibility_document, receiveemployeeid,
      measureemployeeid,
           contact_name, lovidentitytypeid, contact_identity_office,
           contact_identity_number, paymentaccountcontractversid,
           pricelistid, cargocollectid, lovcargostatusid, party_code,
           lovpartnerid, cargonoticeid, trading date, for_worldwide,
           wwcargo_value, planneddeliverydateid,
      actualplanneddeliverydateid,
           plannedarrivaldateid, firstpiecearrivaldateid,
           lastpiecearrivaldateid, deliverydateid, shipmentdateid,
```

```
          canceldateid, auditcreatedateid, auditmodifydateid,
          duedateid, tradingdateid, audit_modify_date2
        )
        select
        null cargono
        , trunc(waybill_Date) OPERATION_DATE
        , w.auditcreateunitid SOURCEUNITID
        , W.DOCUMENT_SERIAL||W.DOCUMENT_NUMBER SERIAL_NUMBER
        , S.SHIPMENT_CODE SHIPMENT_CODE
        , S.SENDERACCOUNTID
        , S.RECEIVERACCOUNTID
        , W.WAYBILLID
        , S.SHIPMENTID
        , W.CARGOID
        , W.CAMPAIGNCONDITIONID
        , W.ACCOUNTCONTRACTVERSIONID
        , decode(nvl(W.WORLDWIDE,0),0,'YURTICI','YURTDISI') YI_YD
        , S.PIECE_COUNT PIECE_COUNT
        , S.TOTAL_VOLUME TOTAL_VOLUME
        , CASE WHEN
W.PAYORACCOUNTCUSTOMERID=HEXTORAW('E56224C0C544734DACA94AC36E23D313')
THEN 'BIREYSEL' ELSE 'KURUMSAL' END BIREYSEL_KURUMSAL
        , C.LOVPAYORTYPEID
        , S.LOVPACKTYPEID
        , W.SERVICEID PRIMARYSERVICEID
        , WP.PRICE TOTAL PRICE
        , WP.INVOICE_PRICE TOTAL_INVOICE_PRICE
        ,
TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(wa
ybill_Date,'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RR
RR'),'DD')  OPERATIONDATEID
        , S.SATELLITEID
        , S.LOVSHIPMENTTYPEID
          , trunc(S.PLANNED_DELIVERY_DATE) PLANNED_DELIVERY_DATE
          , trunc(S.ACTUAL_PLANNED_DELIVERY_DATE)
      ACTUAL_PLANNED_DELIVERY_DATE
          , trunc(S.PLANNED ARRIVAL DATE) PLANNED ARRIVAL DATE
          , trunc(S.FIRST_PIECE_ARRIVAL_DATE) FIRST_PIECE_ARRIVAL_DATE
          , trunc(S.LAST_PIECE_ARRIVAL_DATE) LAST_PIECE_ARRIVAL_DATE
          , S.LOVUNITDISTANCETYPEID, S.DURATION, S.MOBILE,
      S.MANIFESTUNITID
          , S.DESCRIPTION, S.LOVSHIPMENTSTATUSID, S.INTEGRATION_CODE
          , S.CONTENTS_DESCRIPTION, trunc(S.DELIVER_DATE)
      DELIVERY_DATE
          , S.SENDERACCOUNTNAME, S.SENDERACCOUNTADDRESSNAME
          , S.RECEIVERACCOUNTNAME, S.RECEIVERACCOUNTADDRESSNAME,
      S.DELIVERYUNITID
    , S.LOVDELIVERYTYPEID, S.RESPONSIBLEUNITID
    , trunc(S.SHIPMENT DATE) SHIPMENT DATE, S.SHIPMENTDELIVEREDID
    , S.LOVDELIVERYFAILUREREASONID, S.PARENTDELIVERYUNITID
    , S.PARENTSOURCEUNITID, trunc(W.CANCEL_DATE) CANCEL_DATE
    , W.CANCELEDBY, W.CANCELED, W.AUDIT_CREATE_DATE AUDIT_CREATE_DATE
    , W.AUDITCREATEUNITID AUDITCREATEUNITID
    , W.AUDIT_MODIFY_DATE AUDIT_MODIFY_DATE, '0' AUDIT_DELETED
    , W.INVOICEID, W.CANCELUNITID, W.LOVWAYBILLTYPEID,
W.CANCEL_DESCRIPTION
    , W.AUDITCREATEDBY, W.AUDITMODIFIEDBY, W.AUDITMODIFYUNITID
    , W.SENDERACCOUNTADDRESSVERSIONID, W.INVOICEUNITID,
W.COLLECTIONUNITID
```

```
    , W.INVOICEADDRESSVERSIONID, W.WORLDWIDE,
W.LOVDOCUMENTPRINTSTATUSID
    , W.DIFFINVOICEID, W.PAYORACCOUNTCUSTOMERID, W.RETURNED,
W.PUANTUMBONUS
    , W.PUANTUMCANCEL, W.PUANTUMCANCELED, W.PUANTUMCARDNO
    , W.PUANTUMCARDOWNER, W.PUANTUMWEB, trunc(W.DUE_DATE) DUE_DATE
    , W.PAYORACCOUNTNAME, W.INVOICEADDRESSID, W.ACCCONTRACTID
    , W.RECEIVERADDRESSVERSIONID , W.REFCODE, W.TRADING_WAYBILL_NUMBER
    , C.TRADING_GOODS, C.RESPONSIBILITY_DOCUMENT, C.RECEIVEEMPLOYEEID
    , C.MEASUREEMPLOYEEID, C.CONTACT_NAME, C.LOVIDENTITYTYPEID
    , C.CONTACT_IDENTITY_OFFICE, C.CONTACT_IDENTITY_NUMBER
    , C.PAYMENTACCOUNTCONTRACTVERSID
    , C.PRICELISTID, C.CARGOCOLLECTID, C.LOVCARGOSTATUSID,
C.PARTY CODE
    , C.LOVPARTNERID, C.CARGONOTICEID
    , trunc(C.TRADING_DATE) TRADING_DATE, C.FOR_WORLDWIDE,
C.WWCARGO_VALUE
    ,
TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'RRRR')||
TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'MM')||TO
_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'DD')
PLANNEDDELIVERYDATEID
        ,
TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'R
RRR')||TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RR
RR'),'MM')||TO CHAR(TO dATE(trunc(S.ACTUAL PLANNED_DELIVERY_DATE),'DD.
MM.RRRR'),'DD')  ACTUALPLANNEDDELIVERYDATEID
        ,
TO_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')||T
O_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')||TO_C
HAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
PLANNEDARRIVALDATEID
        ,
TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR'
)||TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'MM
')||TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'D
D')  FIRSTPIECEARRIVALDATEID
        ,
TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')
||TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')
||TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
LASTPIECEARRIVALDATEID
        ,
TO_CHAR(TO_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(T
O_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trun
c(S.DELIVER_DATE),'DD.MM.RRRR'),'DD')  DELIVERYDATEID
        ,
TO_CHAR(TO_dATE(trunc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(
TO dATE(trunc(S.SHIPMENT DATE),'DD.MM.RRRR'),'MM')||TO CHAR(TO_dATE(tr
unc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'DD')  SHIPMENTDATEID
        ,
TO_CHAR(TO_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO
_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(
W.CANCEL_DATE),'DD.MM.RRRR'),'DD')  CANCELDATEID
        ,
TO_CHAR(TO_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'RRRR')||TO_C
HAR(TO_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO
 dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'DD')
AUDITCREATEDATEID
```

```
        ,
TO_CHAR(TO_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'RRRR')||TO_C
HAR(TO_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO
_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'DD')
AUDITMODIFYDATEID
        ,
TO_CHAR(TO_dATE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dA
TE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(W.DUE_
DATE),'DD.MM.RRRR'),'DD')  DUEDATEID
        ,
TO_CHAR(TO_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(T
O_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trun
c(C.TRADING_DATE),'DD.MM.RRRR'),'DD')  TRADINGDATEID
        , w.AUDIT_MODIFY_DATE2 AUDIT_MODIFY_DATE2
        from
        waybill w
        inner join waybillprice wp on w.waybillid=WP.WAYBILLID
        inner join shipment s on  s.shipmentid=w.shipmentid
        inner join cargo c on    c.cargoid=w.cargoid
        inner join UNIT U on      U.UNITID=W.AUDITCREATEUNITID
        where  W.AUDIT_MODIFY_DATE2>=p_last_transfer_date
        AND W.AUDIT_MODIFY_DATE2<p_transfer_date
        and W.AUDIT_DELETED='0'
        and WP.AUDIT_DELETED='0'
        and S.AUDIT_DELETED='0'
       ;
    l_rowcount :=l_rowcount+sql%rowcount;
    commit;


--changedamountwaybill
        insert into cargo_fact_tmp
               (
          cargono, operation_date, sourceunitid, serial_number,
          shipment_code, senderaccountid, receiveraccountid,
          waybillid, shipmentid, cargoid, campaignconditionid,
          accountcontractversionid, yi_yd, piece_count, total_volume,
          bireysel_kurumsal, lovpayortypeid, lovpacktypeid,
          primaryserviceid, total_price, total_invoice_price,
          operationdateid, satelliteid, lovshipmenttypeid,
          planned_delivery_date, actual_planned_delivery_date,
          planned_arrival_date, first_piece_arrival_date,
          last_piece_arrival_date, lovunitdistancetypeid,
          duration, mobile, manifestunitid, description,
          lovshipmentstatusid, integration code,
          contents_description, delivery_date, senderaccountname,
          senderaccountaddressname, receiveraccountname,
          receiveraccountaddressname, deliveryunitid,
      lovdeliverytypeid,
          responsibleunitid, shipment_date, shipmentdeliveredid,
          lovdeliveryfailurereasonid, parentdeliveryunitid,
          parentsourceunitid, cancel_date, canceledby, canceled,
          audit_create_date, auditcreateunitid, audit_modify_date,
          audit_deleted, invoiceid, cancelunitid, lovwaybilltypeid,
          cancel_description, auditcreatedby, auditmodifiedby,
          auditmodifyunitid, senderaccountaddressversionid,
          invoiceunitid, collectionunitid, invoiceaddressversionid,
          worldwide, lovdocumentprintstatusid, diffinvoiceid,
```

```
            payoraccountcustomerid, returned, puantumbonus,
            puantumcancel, puantumcanceled, puantumcardno,
            puantumcardowner, puantumweb, due_date, payoraccountname,
            invoiceaddressid, acccontractid, receiveraddressversionid,
            refcode, trading waybill number, trading goods,
            responsibility_document, receiveemployeeid,
        measureemployeeid,
            contact_name, lovidentitytypeid, contact_identity_office,
            contact_identity_number, paymentaccountcontractversid,
            pricelistid, cargocollectid, lovcargostatusid, party_code,
            lovpartnerid, cargonoticeid, trading_date, for_worldwide,
            wwcargo_value, planneddeliverydateid,
        actualplanneddeliverydateid,
            plannedarrivaldateid, firstpiecearrivaldateid,
            lastpiecearrivaldateid, deliverydateid, shipmentdateid,
            canceldateid, auditcreatedateid, auditmodifydateid,
            duedateid, tradingdateid, audit_modify_date2
          )
      select
      null cargono
      , trunc(waybill_Date) OPERATION_DATE
      , w.auditcreateunitid SOURCEUNITID
      , W.DOCUMENT SERIAL||W.DOCUMENT_NUMBER SERIAL_NUMBER
      , S.SHIPMENT_CODE SHIPMENT_CODE
      , S.SENDERACCOUNTID
      , S.RECEIVERACCOUNTID
      , W.WAYBILLID
      , S.SHIPMENTID
      , W.CARGOID
      , W.CAMPAIGNCONDITIONID
      , W.ACCOUNTCONTRACTVERSIONID
      , decode(nvl(W.WORLDWIDE,0),0,'YURTICI','YURTDISI') YI_YD
      , S.PIECE_COUNT PIECE_COUNT
      , S.TOTAL_VOLUME TOTAL_VOLUME
      , CASE WHEN
W.PAYORACCOUNTCUSTOMERID=HEXTORAW('E56224C0C544734DACA94AC36E23D313')
THEN 'BIREYSEL' ELSE 'KURUMSAL' END BIREYSEL_KURUMSAL
      , C.LOVPAYORTYPEID
      , S.LOVPACKTYPEID
      , W.SERVICEID PRIMARYSERVICEID
      , WP.PRICE TOTAL_PRICE
      , WP.INVOICE_PRICE TOTAL_INVOICE_PRICE
      ,
TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(wa
ybill_Date,'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RR
RR'),'DD')  OPERATIONDATEID
      , S.SATELLITEID
      , S.LOVSHIPMENTTYPEID
      , trunc(S.PLANNED DELIVERY DATE) PLANNED_DELIVERY_DATE
      , trunc(S.ACTUAL_PLANNED_DELIVERY_DATE)
ACTUAL_PLANNED_DELIVERY_DATE
      , trunc(S.PLANNED_ARRIVAL_DATE) PLANNED_ARRIVAL_DATE
      , trunc(S.FIRST_PIECE_ARRIVAL_DATE) FIRST_PIECE_ARRIVAL_DATE
      , trunc(S.LAST_PIECE_ARRIVAL_DATE) LAST_PIECE_ARRIVAL_DATE
      , S.LOVUNITDISTANCETYPEID, S.DURATION, S.MOBILE, S.MANIFESTUNITID
      , S.DESCRIPTION, S.LOVSHIPMENTSTATUSID, S.INTEGRATION_CODE
      , S.CONTENTS_DESCRIPTION, trunc(S.DELIVER_DATE) DELIVERY_DATE
      , S.SENDERACCOUNTNAME, S.SENDERACCOUNTADDRESSNAME
```

```
    , S.RECEIVERACCOUNTNAME, S.RECEIVERACCOUNTADDRESSNAME,
S.DELIVERYUNITID
    , S.LOVDELIVERYTYPEID, S.RESPONSIBLEUNITID
    , trunc(S.SHIPMENT_DATE) SHIPMENT_DATE, S.SHIPMENTDELIVEREDID
    , S.LOVDELIVERYFAILUREREASONID, S.PARENTDELIVERYUNITID
    , S.PARENTSOURCEUNITID, trunc(W.CANCEL_DATE) CANCEL_DATE
    , W.CANCELEDBY, W.CANCELED, W.AUDIT_CREATE_DATE AUDIT_CREATE_DATE
    , W.AUDITCREATEUNITID AUDITCREATEUNITID
    , W.AUDIT_MODIFY_DATE AUDIT_MODIFY_DATE, '0' AUDIT_DELETED
    , W.INVOICEID, W.CANCELUNITID, W.LOVWAYBILLTYPEID,
W.CANCEL_DESCRIPTION
    , W.AUDITCREATEDBY, W.AUDITMODIFIEDBY, W.AUDITMODIFYUNITID
    , W.SENDERACCOUNTADDRESSVERSIONID, W.INVOICEUNITID,
W.COLLECTIONUNITID
    , W.INVOICEADDRESSVERSIONID, W.WORLDWIDE,
W.LOVDOCUMENTPRINTSTATUSID
    , W.DIFFINVOICEID, W.PAYORACCOUNTCUSTOMERID, W.RETURNED,
W.PUANTUMBONUS
    , W.PUANTUMCANCEL, W.PUANTUMCANCELED, W.PUANTUMCARDNO
    , W.PUANTUMCARDOWNER, W.PUANTUMWEB, trunc(W.DUE_DATE) DUE_DATE
    , W.PAYORACCOUNTNAME, W.INVOICEADDRESSID, W.ACCCONTRACTID
    , W.RECEIVERADDRESSVERSIONID , W.REFCODE, W.TRADING_WAYBILL_NUMBER
    , C.TRADING GOODS, C.RESPONSIBILITY DOCUMENT, C.RECEIVEEMPLOYEEID
    , C.MEASUREEMPLOYEEID, C.CONTACT_NAME, C.LOVIDENTITYTYPEID
    , C.CONTACT_IDENTITY_OFFICE, C.CONTACT_IDENTITY_NUMBER
    , C.PAYMENTACCOUNTCONTRACTVERSID
    , C.PRICELISTID, C.CARGOCOLLECTID, C.LOVCARGOSTATUSID,
C.PARTY CODE
    , C.LOVPARTNERID, C.CARGONOTICEID
    , trunc(C.TRADING_DATE) TRADING_DATE, C.FOR_WORLDWIDE,
C.WWCARGO_VALUE
    ,
TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'RRRR')||
TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'MM')||TO
_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'DD')
PLANNEDDELIVERYDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'R
RRR')||TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RR
RR'),'MM')||TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.
MM.RRRR'),'DD')  ACTUALPLANNEDDELIVERYDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')||T
O_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')||TO_C
HAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
PLANNEDARRIVALDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR'
)||TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'MM
')||TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'D
D')  FIRSTPIECEARRIVALDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')
||TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')
||TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
LASTPIECEARRIVALDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(T
```

```
O_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trun
c(S.DELIVER_DATE),'DD.MM.RRRR'),'DD')  DELIVERYDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(
TO dATE(trunc(S.SHIPMENT DATE),'DD.MM.RRRR'),'MM')||TO CHAR(TO_dATE(tr
unc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'DD')  SHIPMENTDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO
_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(
W.CANCEL_DATE),'DD.MM.RRRR'),'DD')  CANCELDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'RRRR')||TO_C
HAR(TO_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO
 dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'DD')
AUDITCREATEDATEID
    ,
TO CHAR(TO dATE(trunc(W.AUDIT MODIFY DATE),'DD.MM.RRRR'),'RRRR')||TO C
HAR(TO_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO
_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'DD')
AUDITMODIFYDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dA
TE(trunc(W.DUE DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(W.DUE_
DATE),'DD.MM.RRRR'),'DD')  DUEDATEID
    ,
TO CHAR(TO dATE(trunc(C.TRADING DATE),'DD.MM.RRRR'),'RRRR')||TO CHAR(T
O_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trun
c(C.TRADING DATE),'DD.MM.RRRR'),'DD')  TRADINGDATEID
    , cat.AUDIT_MODIFY_DATE2 AUDIT_MODIFY_DATE2
        from
        waybill w
        inner join waybillprice wp on w.waybillid=WP.WAYBILLID
        inner join shipment s on  s.shipmentid=w.shipmentid
        inner join cargo c on    c.cargoid=w.cargoid
        inner join UNIT U on      U.UNITID=W.AUDITCREATEUNITID
        inner join customeraccounttransfer cat on
CAT.WAYBILLID=W.WAYBILLID
    where  CAT.AUDIT_MODIFY_DATE2 >= p_last_transfer_date
    AND CAT.AUDIT_MODIFY_DATE2<p_transfer_date
    and CAT.AUDIT_DELETED='0'
    and W.AUDIT_DELETED='0'
    and WP.AUDIT_DELETED='0'
    and S.AUDIT_DELETED='0'
    ;
    l_rowcount :=l_rowcount+sql%rowcount;
    commit;

--waybillunitchanced
        insert into cargo_fact_tmp
              (
          cargono, operation_date, sourceunitid, serial_number,
          shipment_code, senderaccountid, receiveraccountid,
          waybillid, shipmentid, cargoid, campaignconditionid,
          accountcontractversionid, yi_yd, piece_count, total_volume,
          bireysel kurumsal, lovpayortypeid, lovpacktypeid,
          primaryserviceid, total_price, total_invoice_price,
          operationdateid, satelliteid, lovshipmenttypeid,
          planned delivery date, actual planned delivery date,
          planned_arrival_date, first_piece_arrival_date,
```

```
            last_piece_arrival_date, lovunitdistancetypeid,
            duration, mobile, manifestunitid, description,
            lovshipmentstatusid, integration_code,
            contents_description, delivery_date, senderaccountname,
            senderaccountaddressname, receiveraccountname,
            receiveraccountaddressname, deliveryunitid,
        lovdeliverytypeid,
            responsibleunitid, shipment_date, shipmentdeliveredid,
            lovdeliveryfailurereasonid, parentdeliveryunitid,
            parentsourceunitid, cancel_date, canceledby, canceled,
            audit_create_date, auditcreateunitid, audit_modify_date,
            audit_deleted, invoiceid, cancelunitid, lovwaybilltypeid,
            cancel_description, auditcreatedby, auditmodifiedby,
            auditmodifyunitid, senderaccountaddressversionid,
            invoiceunitid, collectionunitid, invoiceaddressversionid,
            worldwide, lovdocumentprintstatusid, diffinvoiceid,
            payoraccountcustomerid, returned, puantumbonus,
            puantumcancel, puantumcanceled, puantumcardno,
            puantumcardowner, puantumweb, due_date, payoraccountname,
            invoiceaddressid, acccontractid, receiveraddressversionid,
            refcode, trading_waybill_number, trading_goods,
            responsibility_document, receiveemployeeid,
        measureemployeeid,
            contact_name, lovidentitytypeid, contact_identity_office,
            contact_identity_number, paymentaccountcontractversid,
            pricelistid, cargocollectid, lovcargostatusid, party code,
            lovpartnerid, cargonoticeid, trading_date, for_worldwide,
            wwcargo value, planneddeliverydateid,
        actualplanneddeliverydateid,
            plannedarrivaldateid, firstpiecearrivaldateid,
            lastpiecearrivaldateid, deliverydateid, shipmentdateid,
            canceldateid, auditcreatedateid, auditmodifydateid,
            duedateid, tradingdateid, audit_modify_date2
          )
    select
    null cargono
    , trunc(waybill Date) OPERATION DATE
    , w.auditcreateunitid SOURCEUNITID
    , W.DOCUMENT_SERIAL||W.DOCUMENT_NUMBER SERIAL_NUMBER
    , S.SHIPMENT_CODE SHIPMENT_CODE
    , S.SENDERACCOUNTID
    , S.RECEIVERACCOUNTID
    , W.WAYBILLID
    , S.SHIPMENTID
    , W.CARGOID
    , W.CAMPAIGNCONDITIONID
    , W.ACCOUNTCONTRACTVERSIONID
    , decode(nvl(W.WORLDWIDE,0),0,'YURTICI','YURTDISI') YI_YD
    , S.PIECE COUNT PIECE COUNT
    , S.TOTAL_VOLUME TOTAL_VOLUME
    , CASE WHEN
W.PAYORACCOUNTCUSTOMERID=HEXTORAW('E56224C0C544734DACA94AC36E23D313')
THEN 'BIREYSEL' ELSE 'KURUMSAL' END BIREYSEL_KURUMSAL
    , C.LOVPAYORTYPEID
    , S.LOVPACKTYPEID
    , W.SERVICEID PRIMARYSERVICEID
    , WP.PRICE TOTAL_PRICE
    , WP.INVOICE_PRICE TOTAL_INVOICE_PRICE
```

```
    ,
TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dATE(wa
ybill_Date,'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(waybill_Date,'DD.MM.RR
RR'),'DD')  OPERATIONDATEID
    , S.SATELLITEID
    , S.LOVSHIPMENTTYPEID
    , trunc(S.PLANNED_DELIVERY_DATE) PLANNED_DELIVERY_DATE
    , trunc(S.ACTUAL_PLANNED_DELIVERY_DATE)
ACTUAL_PLANNED_DELIVERY_DATE
    , trunc(S.PLANNED_ARRIVAL_DATE) PLANNED_ARRIVAL_DATE
    , trunc(S.FIRST_PIECE_ARRIVAL_DATE) FIRST_PIECE_ARRIVAL_DATE
    , trunc(S.LAST_PIECE_ARRIVAL_DATE) LAST_PIECE_ARRIVAL_DATE
    , S.LOVUNITDISTANCETYPEID, S.DURATION, S.MOBILE, S.MANIFESTUNITID
    , S.DESCRIPTION, S.LOVSHIPMENTSTATUSID, S.INTEGRATION CODE
    , S.CONTENTS_DESCRIPTION, trunc(S.DELIVER_DATE) DELIVERY_DATE
    , S.SENDERACCOUNTNAME, S.SENDERACCOUNTADDRESSNAME
    , S.RECEIVERACCOUNTNAME, S.RECEIVERACCOUNTADDRESSNAME,
S.DELIVERYUNITID
    , S.LOVDELIVERYTYPEID, S.RESPONSIBLEUNITID
    , trunc(S.SHIPMENT_DATE) SHIPMENT_DATE, S.SHIPMENTDELIVEREDID
    , S.LOVDELIVERYFAILUREREASONID, S.PARENTDELIVERYUNITID
    , S.PARENTSOURCEUNITID, trunc(W.CANCEL_DATE) CANCEL_DATE
    , W.CANCELEDBY, W.CANCELED, W.AUDIT CREATE_DATE AUDIT_CREATE_DATE
    , W.AUDITCREATEUNITID AUDITCREATEUNITID
    , W.AUDIT_MODIFY_DATE AUDIT_MODIFY_DATE, '0' AUDIT_DELETED
    , W.INVOICEID, W.CANCELUNITID, W.LOVWAYBILLTYPEID,
W.CANCEL_DESCRIPTION
    , W.AUDITCREATEDBY, W.AUDITMODIFIEDBY, W.AUDITMODIFYUNITID
    , W.SENDERACCOUNTADDRESSVERSIONID, W.INVOICEUNITID,
W.COLLECTIONUNITID
    , W.INVOICEADDRESSVERSIONID, W.WORLDWIDE,
W.LOVDOCUMENTPRINTSTATUSID
    , W.DIFFINVOICEID, W.PAYORACCOUNTCUSTOMERID, W.RETURNED,
W.PUANTUMBONUS
    , W.PUANTUMCANCEL, W.PUANTUMCANCELED, W.PUANTUMCARDNO
    , W.PUANTUMCARDOWNER, W.PUANTUMWEB, trunc(W.DUE_DATE) DUE_DATE
    , W.PAYORACCOUNTNAME, W.INVOICEADDRESSID, W.ACCCONTRACTID
    , W.RECEIVERADDRESSVERSIONID , W.REFCODE, W.TRADING_WAYBILL_NUMBER
    , C.TRADING_GOODS, C.RESPONSIBILITY_DOCUMENT, C.RECEIVEEMPLOYEEID
    , C.MEASUREEMPLOYEEID, C.CONTACT_NAME, C.LOVIDENTITYTYPEID
    , C.CONTACT_IDENTITY_OFFICE, C.CONTACT_IDENTITY_NUMBER
    , C.PAYMENTACCOUNTCONTRACTVERSID
    , C.PRICELISTID, C.CARGOCOLLECTID, C.LOVCARGOSTATUSID,
C.PARTY_CODE
    , C.LOVPARTNERID, C.CARGONOTICEID
    , trunc(C.TRADING_DATE) TRADING_DATE, C.FOR_WORLDWIDE,
C.WWCARGO_VALUE
    ,
TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'RRRR')||
TO_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'MM')||TO
_CHAR(TO_dATE(trunc(S.PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'DD')
PLANNEDDELIVERYDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RRRR'),'R
RRR')||TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.MM.RR
RR'),'MM')||TO_CHAR(TO_dATE(trunc(S.ACTUAL_PLANNED_DELIVERY_DATE),'DD.
MM.RRRR'),'DD')  ACTUALPLANNEDDELIVERYDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')||T
```

```
O_CHAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')||TO_C
HAR(TO_dATE(trunc(S.PLANNED_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
PLANNEDARRIVALDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR'
)||TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'MM
')||TO_CHAR(TO_dATE(trunc(S.FIRST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'D
D')  FIRSTPIECEARRIVALDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'RRRR')
||TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'MM')
||TO_CHAR(TO_dATE(trunc(S.LAST_PIECE_ARRIVAL_DATE),'DD.MM.RRRR'),'DD')
LASTPIECEARRIVALDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(T
O_dATE(trunc(S.DELIVER_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trun
c(S.DELIVER_DATE),'DD.MM.RRRR'),'DD')  DELIVERYDATEID
    ,
TO_CHAR(TO_dATE(trunc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(
TO_dATE(trunc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(tr
unc(S.SHIPMENT_DATE),'DD.MM.RRRR'),'DD')  SHIPMENTDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO
_dATE(trunc(W.CANCEL_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(
W.CANCEL_DATE),'DD.MM.RRRR'),'DD')  CANCELDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'RRRR')||TO_C
HAR(TO_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO
_dATE(trunc(W.AUDIT_CREATE_DATE),'DD.MM.RRRR'),'DD')
AUDITCREATEDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'RRRR')||TO_C
HAR(TO_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO
_dATE(trunc(W.AUDIT_MODIFY_DATE),'DD.MM.RRRR'),'DD')
AUDITMODIFYDATEID
    ,
TO_CHAR(TO_dATE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(TO_dA
TE(trunc(W.DUE_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trunc(W.DUE_
DATE),'DD.MM.RRRR'),'DD')  DUEDATEID
    ,
TO_CHAR(TO_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'RRRR')||TO_CHAR(T
O_dATE(trunc(C.TRADING_DATE),'DD.MM.RRRR'),'MM')||TO_CHAR(TO_dATE(trun
c(C.TRADING_DATE),'DD.MM.RRRR'),'DD')  TRADINGDATEID
    , WCR.AUDIT_MODIFY_DATE2 AUDIT_MODIFY_DATE2
        from
       waybill w
       inner join waybillprice wp on w.waybillid=WP.WAYBILLID
       inner join shipment s on  s.shipmentid=w.shipmentid
       inner join cargo c on    c.cargoid=w.cargoid
       inner join UNIT U on      U.UNITID=W.AUDITCREATEUNITID
       inner join waybillchangerevenueunit wcr on
wcr.WAYBILLID=W.WAYBILLID
    where  WCR.AUDIT_MODIFY_DATE2 >= p_last_transfer_date
    AND WCR.AUDIT_MODIFY_DATE2<p_transfer_date
    and wcr.audit Deleted='0'
    and WCR.AUDIT_DELETED='0'
    and W.AUDIT_DELETED='0'
    and WP.AUDIT_DELETED='0'
    and S.AUDIT_DELETED='0'
```

```
    ;
    l_rowcount :=l_rowcount+sql%rowcount;

    update  whtransfer
    set row count = l rowcount
    where whtransferid=p_whtransferid;

    commit;


    PROC_SEND_MAIL('mertsun@araskargo.com.tr','warehouse-cargo: ' ||
p_transfer_date,'Tempe cargo data aktarimi tamamlandı');


    declare
    l_whtransferid varchar2(32);
    l table_name varchar2(100):='CARGO_FACT';
    BEGIN
    l_whtransferid :=to_char(p_whtransferid);
    delete from t_transfer@warehouse where table_name=l_table_name;
    insert into t_transfer@warehouse(table_name,whtransferid)
values(l_table_name, l_whtransferid);
    commit;


    ms_prc :=  DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@warehouse
        ('exec msdb.dbo.sp_start_job ''JB_TRANS_CARGO_FACT''');
    commit;
    END;

    update  whtransfer
    set lovwhtransferstatusid=2,TRANSFER_COMPLETE_DATE=sysdate
    where whtransferid=p_whtransferid;
    commit;

    PROC_SEND_MAIL('mertsun@araskargo.com.tr','warehouse-cargo: ' ||
p_transfer_date,'warehousea '|| p_transfer_date || ' tarihine kadar
tutari degisen cargo datasi aktarildi');

END PROC_wh_cargo;
/
```

APPENDIX B

Transact/SQL Source code of ETL–Loading Phase Procedure of Cargo Fact

```
USE [ARASWH]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:        Mert SUN
-- Create date: 25.06.2011
-- Description:   Importing CARGO_FACT table
-- =============================================

CREATE PROCEDURE [dbo].[PROC_TRANS_CARGO_FACT]
AS
BEGIN

SET NOCOUNT ON

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED


declare @l_whtransferid nvarchar(32)
declare @l_table_name nvarchar(32) ='CARGO_FACT'
SELECT @l_whtransferid = whtransferid
FROM t_transfer
WHERE table_name = @l_table_name



--truncate sql tmp
/*TRUNCATE ET VE vs*/
truncate table [ARASWH].[dbo].[CARGO_FACT_TMP];

INSERT INTO [ARASWH].[dbo].[CARGO_FACT_TMP]
           (CARGONO,            OPERATION_DATE,            SOURCEUNITID,
      SERIAL_NUMBER,SHIPMENT_CODE
           ,SENDERACCOUNTIDRECEIVERACCOUNTID,      WAYBILLIDSHIPMENTID,
CARGOID
           ,CAMPAIGNCONDITIONID, ACCOUNTCONTRACTVERSIONID, YI_YD
           ,PIECE_COUNT,                TOTAL_VOLUME,BIREYSEL_KURUMSAL,
PRIMARYSERVICEID
           ,TOTAL_PRICE      ,TOTAL_INVOICE_PRICE,      OPERATIONDATEID,
SATELLITEID
           ,PLANNED_DELIVERY_DATE, ACTUAL_PLANNED_DELIVERY_DATE
           ,PLANNED_ARRIVAL_DATE, FIRST_PIECE_ARRIVAL_DATE
```

```
,LAST_PIECE_ARRIVAL_DATE,DURATION,MOBILE,MANIFESTUNITID,DESCRIPTION
           ,INTEGRATION_CODE, CONTENTS_DESCRIPTION, DELIVERY_DATE
           ,SENDERACCOUNTNAME,                SENDERACCOUNTADDRESSNAME,
RECEIVERACCOUNTNAME
           ,RECEIVERACCOUNTADDRESSNAME, INVOICEID, CANCELUNITID
           ,LOVWAYBILLTYPEID, CANCEL_DESCRIPTION, AUDITCREATEDBY
           ,AUDITMODIFIEDBYAUDITMODIFYUNITID,
SENDERACCOUNTADDRESSVERSIONID
           ,INVOICEUNITID, COLLECTIONUNITID, INVOICEADDRESSVERSIONID
           ,WORLDWIDE, LOVDOCUMENTPRINTSTATUSID, DIFFINVOICEID
           ,PAYORACCOUNTCUSTOMERID,       RETURNED,       PUANTUMBONUS,
PUANTUMCANCEL
           ,PUANTUMCANCELED,      PUANTUMCARDNO,      PUANTUMCARDOWNER,
PUANTUMWEB
           ,DUE_DATE, PAYORACCOUNTNAME, INVOICEADDRESSID
           ,ACCCONTRACTID, RECEIVERADDRESSVERSIONID, REFCODE
           ,TRADING_WAYBILL_NUMBER                        ,TRADING_GOODS
,RESPONSIBILITY_DOCUMENT
           ,RECEIVEEMPLOYEEID, MEASUREEMPLOYEEID, CONTACT_NAME
           ,LOVIDENTITYTYPEID,CONTACT_IDENTITY_OFFICE,
CONTACT_IDENTITY_NUMBER
           ,PAYMENTACCOUNTCONTRACTVERSID, PRICELISTID, CARGOCOLLECTID
           ,LOVCARGOSTATUSID, PARTY_CODE, LOVPARTNERID, CARGONOTICEID
           ,TRADING_DATE, FOR_WORLDWIDE, WWCARGO_VALUE, DELIVERYUNITID
           ,RESPONSIBLEUNITID, SHIPMENT_DATE, SHIPMENTDELIVEREDID
           ,PARENTDELIVERYUNITID, PARENTSOURCEUNITID, PRINTERBARCODEID
           ,LOVPAYORTYPEID, LOVPACKTYPEID, LOVSHIPMENTTYPEID
           ,LOVUNITDISTANCETYPEID,                LOVSHIPMENTSTATUSID,
LOVDELIVERYTYPEID
           ,LOVDELIVERYFAILUREREASONID,    CANCEL_DATE,    CANCELEDBY,
CANCELED
           ,AUDIT_CREATE_DATE, AUDITCREATEUNITID, AUDIT_MODIFY_DATE
           ,AUDIT_DELETED,                      PLANNEDDELIVERYDATEID,
ACTUALPLANNEDDELIVERYDATEID
           ,PLANNEDARRIVALDATEID, FIRSTPIECEARRIVALDATEID
           ,LASTPIECEARRIVALDATEID, DELIVERYDATEID, SHIPMENTDATEID
           ,CANCELDATEID,    AUDITCREATEDATEID,    AUDITMODIFYDATEID,
DUEDATEID
           ,TRADINGDATEID,         CARGO_CODE,         SENDERACCOUNTNO,
RECEIVERACCOUNTNO
           ,SENDERACCOUNTADDRESSVERSIONNO, INVOICEADDRESSVERSIONNO
           ,INVOICEADDRESSNO, RECEIVERADDRESSVERSIONNO
           ,WH_TRANSFER_DATE, AUDIT_MODIFY_DATE2)
SELECT
            CARGONO,           OPERATION_DATE,           SOURCEUNITID,
     SERIAL_NUMBER,SHIPMENT_CODE
           ,SENDERACCOUNTIDRECEIVERACCOUNTID,      WAYBILLIDSHIPMENTID,
CARGOID
           ,CAMPAIGNCONDITIONID, ACCOUNTCONTRACTVERSIONID, YI_YD
           ,PIECE_COUNT,               TOTAL_VOLUME,BIREYSEL_KURUMSAL,
PRIMARYSERVICEID
           ,TOTAL_PRICE     ,TOTAL_INVOICE_PRICE,     OPERATIONDATEID,
SATELLITEID
           ,PLANNED_DELIVERY_DATE, ACTUAL_PLANNED_DELIVERY_DATE
           ,PLANNED_ARRIVAL_DATE, FIRST_PIECE_ARRIVAL_DATE

,LAST_PIECE_ARRIVAL_DATE,DURATION,MOBILE,MANIFESTUNITID,DESCRIPTION
           ,INTEGRATION_CODE, CONTENTS_DESCRIPTION, DELIVERY_DATE
```

```
           ,SENDERACCOUNTNAME,                SENDERACCOUNTADDRESSNAME,
RECEIVERACCOUNTNAME
           ,RECEIVERACCOUNTADDRESSNAME, INVOICEID, CANCELUNITID
           ,LOVWAYBILLTYPEID, CANCEL_DESCRIPTION, AUDITCREATEDBY
           ,AUDITMODIFIEDBYAUDITMODIFYUNITID,
SENDERACCOUNTADDRESSVERSIONID
           ,INVOICEUNITID, COLLECTIONUNITID, INVOICEADDRESSVERSIONID
           ,WORLDWIDE, LOVDOCUMENTPRINTSTATUSID, DIFFINVOICEID
           ,PAYORACCOUNTCUSTOMERID,       RETURNED,       PUANTUMBONUS,
PUANTUMCANCEL
           ,PUANTUMCANCELED,      PUANTUMCARDNO,      PUANTUMCARDOWNER,
PUANTUMWEB
           ,DUE_DATE, PAYORACCOUNTNAME, INVOICEADDRESSID
           ,ACCCONTRACTID, RECEIVERADDRESSVERSIONID, REFCODE
           ,TRADING_WAYBILL_NUMBER                        ,TRADING_GOODS
,RESPONSIBILITY_DOCUMENT
           ,RECEIVEEMPLOYEEID, MEASUREEMPLOYEEID, CONTACT_NAME
           ,LOVIDENTITYTYPEID,CONTACT_IDENTITY_OFFICE,
CONTACT_IDENTITY_NUMBER
           ,PAYMENTACCOUNTCONTRACTVERSID, PRICELISTID, CARGOCOLLECTID
           ,LOVCARGOSTATUSID, PARTY_CODE, LOVPARTNERID, CARGONOTICEID
           ,TRADING_DATE, FOR_WORLDWIDE, WWCARGO_VALUE, DELIVERYUNITID
           ,RESPONSIBLEUNITID, SHIPMENT_DATE, SHIPMENTDELIVEREDID
           ,PARENTDELIVERYUNITID, PARENTSOURCEUNITID, PRINTERBARCODEID
           ,LOVPAYORTYPEID, LOVPACKTYPEID, LOVSHIPMENTTYPEID
           ,LOVUNITDISTANCETYPEID,                LOVSHIPMENTSTATUSID,
LOVDELIVERYTYPEID
           ,LOVDELIVERYFAILUREREASONID,     CANCEL_DATE,     CANCELEDBY,
CANCELED
           ,AUDIT_CREATE_DATE, AUDITCREATEUNITID, AUDIT_MODIFY_DATE
           ,AUDIT_DELETED,                      PLANNEDDELIVERYDATEID,
ACTUALPLANNEDDELIVERYDATEID
           ,PLANNEDARRIVALDATEID, FIRSTPIECEARRIVALDATEID
           ,LASTPIECEARRIVALDATEID, DELIVERYDATEID, SHIPMENTDATEID
           ,CANCELDATEID,     AUDITCREATEDATEID,     AUDITMODIFYDATEID,
DUEDATEID
           ,TRADINGDATEID,         CARGO_CODE,         SENDERACCOUNTNO,
RECEIVERACCOUNTNO
           ,SENDERACCOUNTADDRESSVERSIONNO, INVOICEADDRESSVERSIONNO
           ,INVOICEADDRESSNO, RECEIVERADDRESSVERSIONNO
           ,WH_TRANSFER_DATE, AUDIT_MODIFY_DATE2
   FROM
      openquery(ESASLIVE, '
      select
      *
      from (select
      ROW_NUMBER()    over    (partition    by    WAYBILLID    order    by
      audit_modify_date2 desc) rn,
      CARGONO,
      CASE    WHEN    to_char(OPERATION_DATE,    ''YYYY'')<1000    then
      to_date(''1901'',''YYYY'')      ELSE      OPERATION_DATE      END
      OPERATION_DATE,
      SOURCEUNITID,
      SERIAL_NUMBER,
      SHIPMENT_CODE,
      CAST (SENDERACCOUNTID AS VARCHAR2(32)) SENDERACCOUNTID,
      CAST (RECEIVERACCOUNTID AS VARCHAR2(32)) RECEIVERACCOUNTID,
      CAST (WAYBILLID AS VARCHAR2(32)) WAYBILLID,
      CAST (SHIPMENTID AS VARCHAR2(32)) SHIPMENTID,
```

```
CAST (CARGOID AS VARCHAR2(32)) CARGOID,
CAST (CAMPAIGNCONDITIONID AS VARCHAR2(32)) CAMPAIGNCONDITIONID,
CAST       (ACCOUNTCONTRACTVERSIONID       AS       VARCHAR2(32))
ACCOUNTCONTRACTVERSIONID,
YI_YD,
PIECE_COUNT,
TOTAL_VOLUME,
BIREYSEL_KURUMSAL,
CAST (PRIMARYSERVICEID AS VARCHAR2(32)) PRIMARYSERVICEID,
TOTAL_PRICE,
TOTAL_INVOICE_PRICE,
OPERATIONDATEID,
CAST (SATELLITEID AS VARCHAR2(32)) SATELLITEID,
CASE  WHEN  to_char(PLANNED_DELIVERY_DATE,  ''YYYY'')<1000  then
to_date(''1901'',''YYYY'')   ELSE   PLANNED_DELIVERY_DATE   END
PLANNED_DELIVERY_DATE,
CASE  WHEN  to_char(ACTUAL_PLANNED_DELIVERY_DATE,  ''YYYY'')<1000
then               to_date(''1901'',''YYYY'')               ELSE
ACTUAL_PLANNED_DELIVERY_DATE END  ACTUAL_PLANNED_DELIVERY_DATE,
CASE  WHEN  to_char(PLANNED_ARRIVAL_DATE,  ''YYYY'')<1000  then
to_date(''1901'',''YYYY'')   ELSE   PLANNED_ARRIVAL_DATE   END
PLANNED_ARRIVAL_DATE,
CASE WHEN to_char(FIRST_PIECE_ARRIVAL_DATE, ''YYYY'')<1000 then
to_date(''1901'',''YYYY'')  ELSE  FIRST_PIECE_ARRIVAL_DATE  END
FIRST_PIECE_ARRIVAL_DATE,
CASE WHEN to_char(LAST_PIECE_ARRIVAL_DATE, ''YYYY'')<1000 then
to_date(''1901'',''YYYY'')  ELSE  LAST_PIECE_ARRIVAL_DATE  END
LAST_PIECE_ARRIVAL_DATE,
DURATION,
MOBILE,
MANIFESTUNITID,
DESCRIPTION,
INTEGRATION_CODE,
CONTENTS_DESCRIPTION,
CASE    WHEN    to_char(DELIVERY_DATE,    ''YYYY'')<1000    then
to_date(''1901'',''YYYY'')      ELSE      DELIVERY_DATE      END
DELIVERY_DATE,
SENDERACCOUNTNAME,
SENDERACCOUNTADDRESSNAME,
RECEIVERACCOUNTNAME,
RECEIVERACCOUNTADDRESSNAME,
CAST (INVOICEID AS VARCHAR2(32)) INVOICEID,
CANCELUNITID,
LOVWAYBILLTYPEID,
CANCEL_DESCRIPTION,
AUDITCREATEDBY,
AUDITMODIFIEDBY,
AUDITMODIFYUNITID,
CAST     (SENDERACCOUNTADDRESSVERSIONID     AS     VARCHAR2(32))
SENDERACCOUNTADDRESSVERSIONID,
INVOICEUNITID,
COLLECTIONUNITID,
CAST       (INVOICEADDRESSVERSIONID       AS       VARCHAR2(32))
INVOICEADDRESSVERSIONID,
WORLDWIDE,
LOVDOCUMENTPRINTSTATUSID,
CAST (DIFFINVOICEID AS VARCHAR2(32)) DIFFINVOICEID,
CAST       (PAYORACCOUNTCUSTOMERID       AS       VARCHAR2(32))
PAYORACCOUNTCUSTOMERID,
```

```
RETURNED,
PUANTUMBONUS,
PUANTUMCANCEL,
PUANTUMCANCELED,
PUANTUMCARDNO,
PUANTUMCARDOWNER,
PUANTUMWEB,
CASE      WHEN      to_char(DUE_DATE,      ''YYYY'')<1000      then
to_date(''1901'',''YYYY'') ELSE DUE_DATE END  DUE_DATE,
PAYORACCOUNTNAME,
CAST (INVOICEADDRESSID AS VARCHAR2(32)) INVOICEADDRESSID,
CAST (ACCCONTRACTID AS VARCHAR2(32)) ACCCONTRACTID,
CAST       (RECEIVERADDRESSVERSIONID       AS       VARCHAR2(32))
RECEIVERADDRESSVERSIONID,
REFCODE,
TRADING_WAYBILL_NUMBER,
TRADING_GOODS,
RESPONSIBILITY_DOCUMENT,
RECEIVEEMPLOYEEID,
MEASUREEMPLOYEEID,
CONTACT_NAME,
LOVIDENTITYTYPEID,
CONTACT_IDENTITY_OFFICE,
CONTACT_IDENTITY_NUMBER,
CAST      (PAYMENTACCOUNTCONTRACTVERSID      AS      VARCHAR2(32))
PAYMENTACCOUNTCONTRACTVERSID,
CAST (PRICELISTID AS VARCHAR2(32)) PRICELISTID,
CAST (CARGOCOLLECTID AS VARCHAR2(32)) CARGOCOLLECTID,
LOVCARGOSTATUSID,
PARTY_CODE,
LOVPARTNERID,
CAST (CARGONOTICEID AS VARCHAR2(32)) CARGONOTICEID,
CASE     WHEN     to_char(TRADING_DATE,     ''YYYY'')<1000     then
to_date(''1901'',''YYYY'') ELSE TRADING_DATE END  TRADING_DATE,
FOR_WORLDWIDE,
WWCARGO_VALUE,
DELIVERYUNITID,
RESPONSIBLEUNITID,
CASE     WHEN     to_char(SHIPMENT_DATE,     ''YYYY'')<1000     then
to_date(''1901'',''YYYY'')       ELSE       SHIPMENT_DATE       END
SHIPMENT_DATE,
CAST (SHIPMENTDELIVEREDID AS VARCHAR2(32)) SHIPMENTDELIVEREDID,
PARENTDELIVERYUNITID,
PARENTSOURCEUNITID,
CAST (PRINTERBARCODEID AS VARCHAR2(32)) PRINTERBARCODEID,
LOVPAYORTYPEID,
LOVPACKTYPEID,
LOVSHIPMENTTYPEID,
LOVUNITDISTANCETYPEID,
LOVSHIPMENTSTATUSID,
LOVDELIVERYTYPEID,
LOVDELIVERYFAILUREREASONID,
CASE     WHEN     to_char(CANCEL_DATE,     ''YYYY'')<1000     then
to_date(''1901'',''YYYY'') ELSE CANCEL_DATE END  CANCEL_DATE,
CANCELEDBY,
CANCELED,
CASE    WHEN    to_char(AUDIT_CREATE_DATE,    ''YYYY'')<1000    then
to_date(''1901'',''YYYY'')      ELSE      AUDIT_CREATE_DATE      END
AUDIT_CREATE_DATE,
```

```
        AUDITCREATEUNITID,
        CASE    WHEN    to_char(AUDIT_MODIFY_DATE,    ''YYYY'')<1000    then
        to_date(''1901'',''YYYY'')      ELSE      AUDIT_MODIFY_DATE      END
        AUDIT_MODIFY_DATE,
        AUDIT_DELETED,
        PLANNEDDELIVERYDATEID,
        ACTUALPLANNEDDELIVERYDATEID,
        PLANNEDARRIVALDATEID,
        FIRSTPIECEARRIVALDATEID,
        LASTPIECEARRIVALDATEID,
        DELIVERYDATEID,
        SHIPMENTDATEID,
        CANCELDATEID,
        AUDITCREATEDATEID,
        AUDITMODIFYDATEID,
        DUEDATEID,
        TRADINGDATEID,
        CARGO_CODE,
        SENDERACCOUNTNO,
        RECEIVERACCOUNTNO,
        SENDERACCOUNTADDRESSVERSIONNO,
        INVOICEADDRESSVERSIONNO,
        INVOICEADDRESSNO,
        RECEIVERADDRESSVERSIONNO,
        CASE    WHEN    to_char(WH_TRANSFER_DATE,    ''YYYY'')<1000    then
        to_date(''1901'',''YYYY'')      ELSE      WH_TRANSFER_DATE      END
        WH_TRANSFER_DATE,
        CASE    WHEN    to_char(AUDIT_MODIFY_DATE2,    ''YYYY'')<1000    then
        to_date(''1901'',''YYYY'')      ELSE      AUDIT_MODIFY_DATE2      END
        AUDIT_MODIFY_DATE2 from CARGO_FACT_TMP)
        where rn=1
        ');

        MERGE [ARASWH].[dbo].[CARGO_FACT] as stm USING ( SELECT
        OPERATION_DATE, SOURCEUNITID, SERIAL_NUMBER,SHIPMENT_CODE
             ,SENDERACCOUNTIDRECEIVERACCOUNTID,      WAYBILLIDSHIPMENTID,
CARGOID
             ,CAMPAIGNCONDITIONID, ACCOUNTCONTRACTVERSIONID, YI_YD
             ,PIECE_COUNT,                TOTAL_VOLUME,BIREYSEL_KURUMSAL,
PRIMARYSERVICEID
             ,TOTAL_PRICE      ,TOTAL_INVOICE_PRICE,      OPERATIONDATEID,
SATELLITEID
             ,PLANNED_DELIVERY_DATE, ACTUAL_PLANNED_DELIVERY_DATE
             ,PLANNED_ARRIVAL_DATE, FIRST_PIECE_ARRIVAL_DATE

,LAST_PIECE_ARRIVAL_DATE,DURATION,MOBILE,MANIFESTUNITID,DESCRIPTION
             ,INTEGRATION_CODE, CONTENTS_DESCRIPTION, DELIVERY_DATE
             ,SENDERACCOUNTNAME,                SENDERACCOUNTADDRESSNAME,
RECEIVERACCOUNTNAME
             ,RECEIVERACCOUNTADDRESSNAME, INVOICEID, CANCELUNITID
             ,LOVWAYBILLTYPEID, CANCEL_DESCRIPTION, AUDITCREATEDBY
             ,AUDITMODIFIEDBYAUDITMODIFYUNITID,
SENDERACCOUNTADDRESSVERSIONID
             ,INVOICEUNITID, COLLECTIONUNITID, INVOICEADDRESSVERSIONID
             ,WORLDWIDE, LOVDOCUMENTPRINTSTATUSID, DIFFINVOICEID
             ,PAYORACCOUNTCUSTOMERID,       RETURNED,       PUANTUMBONUS,
PUANTUMCANCEL
             ,PUANTUMCANCELED,      PUANTUMCARDNO,      PUANTUMCARDOWNER,
PUANTUMWEB
```

```
            ,DUE_DATE, PAYORACCOUNTNAME, INVOICEADDRESSID
            ,ACCCONTRACTID, RECEIVERADDRESSVERSIONID, REFCODE
            ,TRADING_WAYBILL_NUMBER                         ,TRADING_GOODS
,RESPONSIBILITY_DOCUMENT
            ,RECEIVEEMPLOYEEID, MEASUREEMPLOYEEID, CONTACT_NAME
            ,LOVIDENTITYTYPEID,CONTACT_IDENTITY_OFFICE,
CONTACT_IDENTITY_NUMBER
            ,PAYMENTACCOUNTCONTRACTVERSID, PRICELISTID, CARGOCOLLECTID
            ,LOVCARGOSTATUSID, PARTY_CODE, LOVPARTNERID, CARGONOTICEID
            ,TRADING_DATE, FOR_WORLDWIDE, WWCARGO_VALUE, DELIVERYUNITID
            ,RESPONSIBLEUNITID, SHIPMENT_DATE, SHIPMENTDELIVEREDID
            ,PARENTDELIVERYUNITID, PARENTSOURCEUNITID, PRINTERBARCODEID
            ,LOVPAYORTYPEID, LOVPACKTYPEID, LOVSHIPMENTTYPEID
            ,LOVUNITDISTANCETYPEID,                 LOVSHIPMENTSTATUSID,
LOVDELIVERYTYPEID
            ,LOVDELIVERYFAILUREREASONID,    CANCEL_DATE,    CANCELEDBY,
CANCELED
            ,AUDIT_CREATE_DATE, AUDITCREATEUNITID, AUDIT_MODIFY_DATE
            ,AUDIT_DELETED,                         PLANNEDDELIVERYDATEID,
ACTUALPLANNEDDELIVERYDATEID
            ,PLANNEDARRIVALDATEID, FIRSTPIECEARRIVALDATEID
            ,LASTPIECEARRIVALDATEID, DELIVERYDATEID, SHIPMENTDATEID
            ,CANCELDATEID,    AUDITCREATEDATEID,    AUDITMODIFYDATEID,
DUEDATEID
            ,TRADINGDATEID,         CARGO_CODE,         SENDERACCOUNTNO,
RECEIVERACCOUNTNO
            ,SENDERACCOUNTADDRESSVERSIONNO, INVOICEADDRESSVERSIONNO
            ,INVOICEADDRESSNO, RECEIVERADDRESSVERSIONNO
            ,WH_TRANSFER_DATE, AUDIT_MODIFY_DATE2
      FROM dbo.CARGO_FACT_TMP
      ) as sd ON  stm.WAYBILLID= sd.WAYBILLID
       WHEN matched then UPDATE SET
      stm.OPERATION_DATE = sd.OPERATION_DATE,
      stm.SOURCEUNITID = sd.SOURCEUNITID,
      stm.SERIAL_NUMBER = sd.SERIAL_NUMBER,
      stm.SHIPMENT_CODE = sd.SHIPMENT_CODE,
      stm.SENDERACCOUNTID = sd.SENDERACCOUNTID,
      stm.RECEIVERACCOUNTID = sd.RECEIVERACCOUNTID,
      stm.SHIPMENTID = sd.SHIPMENTID,
      stm.CARGOID = sd.CARGOID,
      stm.CAMPAIGNCONDITIONID = sd.CAMPAIGNCONDITIONID,
      stm.ACCOUNTCONTRACTVERSIONID = sd.ACCOUNTCONTRACTVERSIONID,
      stm.YI_YD = sd.YI_YD,
      stm.PIECE_COUNT = sd.PIECE_COUNT,
      stm.TOTAL_VOLUME = sd.TOTAL_VOLUME,
      stm.BIREYSEL_KURUMSAL = sd.BIREYSEL_KURUMSAL,
      stm.PRIMARYSERVICEID = sd.PRIMARYSERVICEID,
      stm.TOTAL_PRICE = sd.TOTAL_PRICE,
      stm.TOTAL_INVOICE_PRICE = sd.TOTAL_INVOICE_PRICE,
      stm.OPERATIONDATEID = sd.OPERATIONDATEID,
      stm.SATELLITEID = sd.SATELLITEID,
      stm.PLANNED_DELIVERY_DATE = sd.PLANNED_DELIVERY_DATE,
      stm.ACTUAL_PLANNED_DELIVERY_DATE                                  =
      sd.ACTUAL_PLANNED_DELIVERY_DATE,
      stm.PLANNED_ARRIVAL_DATE = sd.PLANNED_ARRIVAL_DATE,
      stm.FIRST_PIECE_ARRIVAL_DATE = sd.FIRST_PIECE_ARRIVAL_DATE,
      stm.LAST_PIECE_ARRIVAL_DATE = sd.LAST_PIECE_ARRIVAL_DATE,
      stm.DURATION = sd.DURATION,
      stm.MOBILE = sd.MOBILE,
```

```
stm.MANIFESTUNITID = sd.MANIFESTUNITID,
stm.DESCRIPTION = sd.DESCRIPTION,
stm.INTEGRATION_CODE = sd.INTEGRATION_CODE,
stm.CONTENTS_DESCRIPTION = sd.CONTENTS_DESCRIPTION,
stm.DELIVERY_DATE = sd.DELIVERY_DATE,
stm.SENDERACCOUNTNAME = sd.SENDERACCOUNTNAME,
stm.SENDERACCOUNTADDRESSNAME = sd.SENDERACCOUNTADDRESSNAME,
stm.RECEIVERACCOUNTNAME = sd.RECEIVERACCOUNTNAME,
stm.RECEIVERACCOUNTADDRESSNAME = sd.RECEIVERACCOUNTADDRESSNAME,
stm.INVOICEID = sd.INVOICEID,
stm.CANCELUNITID = sd.CANCELUNITID,
stm.LOVWAYBILLTYPEID = sd.LOVWAYBILLTYPEID,
stm.CANCEL_DESCRIPTION = sd.CANCEL_DESCRIPTION,
stm.AUDITCREATEDBY = sd.AUDITCREATEDBY,
stm.AUDITMODIFIEDBY = sd.AUDITMODIFIEDBY,
stm.AUDITMODIFYUNITID = sd.AUDITMODIFYUNITID,
stm.SENDERACCOUNTADDRESSVERSIONID                                    =
sd.SENDERACCOUNTADDRESSVERSIONID,
stm.INVOICEUNITID = sd.INVOICEUNITID,
stm.COLLECTIONUNITID = sd.COLLECTIONUNITID,
stm.INVOICEADDRESSVERSIONID = sd.INVOICEADDRESSVERSIONID,
stm.WORLDWIDE = sd.WORLDWIDE,
stm.LOVDOCUMENTPRINTSTATUSID = sd.LOVDOCUMENTPRINTSTATUSID,
stm.DIFFINVOICEID = sd.DIFFINVOICEID,
stm.PAYORACCOUNTCUSTOMERID = sd.PAYORACCOUNTCUSTOMERID,
stm.RETURNED = sd.RETURNED,
stm.PUANTUMBONUS = sd.PUANTUMBONUS,
stm.PUANTUMCANCEL = sd.PUANTUMCANCEL,
stm.PUANTUMCANCELED = sd.PUANTUMCANCELED,
stm.PUANTUMCARDNO = sd.PUANTUMCARDNO,
stm.PUANTUMCARDOWNER = sd.PUANTUMCARDOWNER,
stm.PUANTUMWEB = sd.PUANTUMWEB,
stm.DUE_DATE = sd.DUE_DATE,
stm.PAYORACCOUNTNAME = sd.PAYORACCOUNTNAME,
stm.INVOICEADDRESSID = sd.INVOICEADDRESSID,
stm.ACCCONTRACTID = sd.ACCCONTRACTID,
stm.RECEIVERADDRESSVERSIONID = sd.RECEIVERADDRESSVERSIONID,
stm.REFCODE = sd.REFCODE,
stm.TRADING_WAYBILL_NUMBER = sd.TRADING_WAYBILL_NUMBER,
stm.TRADING_GOODS = sd.TRADING_GOODS,
stm.RESPONSIBILITY_DOCUMENT = sd.RESPONSIBILITY_DOCUMENT,
stm.RECEIVEEMPLOYEEID = sd.RECEIVEEMPLOYEEID,
stm.MEASUREEMPLOYEEID = sd.MEASUREEMPLOYEEID,
stm.CONTACT_NAME = sd.CONTACT_NAME,
stm.LOVIDENTITYTYPEID = sd.LOVIDENTITYTYPEID,
stm.CONTACT_IDENTITY_OFFICE = sd.CONTACT_IDENTITY_OFFICE,
stm.CONTACT_IDENTITY_NUMBER = sd.CONTACT_IDENTITY_NUMBER,
stm.PAYMENTACCOUNTCONTRACTVERSID                                     =
sd.PAYMENTACCOUNTCONTRACTVERSID,
stm.PRICELISTID = sd.PRICELISTID,
stm.CARGOCOLLECTID = sd.CARGOCOLLECTID,
stm.LOVCARGOSTATUSID = sd.LOVCARGOSTATUSID,
stm.PARTY_CODE = sd.PARTY_CODE,
stm.LOVPARTNERID = sd.LOVPARTNERID,
stm.CARGONOTICEID = sd.CARGONOTICEID,
stm.TRADING_DATE = sd.TRADING_DATE,
stm.FOR_WORLDWIDE = sd.FOR_WORLDWIDE,
stm.WWCARGO_VALUE = sd.WWCARGO_VALUE,
stm.DELIVERYUNITID = sd.DELIVERYUNITID,
```

```
        stm.RESPONSIBLEUNITID = sd.RESPONSIBLEUNITID,
        stm.SHIPMENT_DATE = sd.SHIPMENT_DATE,
        stm.SHIPMENTDELIVEREDID = sd.SHIPMENTDELIVEREDID,
        stm.PARENTDELIVERYUNITID = sd.PARENTDELIVERYUNITID,
        stm.PARENTSOURCEUNITID = sd.PARENTSOURCEUNITID,
        stm.PRINTERBARCODEID = sd.PRINTERBARCODEID,
        stm.LOVPAYORTYPEID = sd.LOVPAYORTYPEID,
        stm.LOVPACKTYPEID = sd.LOVPACKTYPEID,
        stm.LOVSHIPMENTTYPEID = sd.LOVSHIPMENTTYPEID,
        stm.LOVUNITDISTANCETYPEID = sd.LOVUNITDISTANCETYPEID,
        stm.LOVSHIPMENTSTATUSID = sd.LOVSHIPMENTSTATUSID,
        stm.LOVDELIVERYTYPEID = sd.LOVDELIVERYTYPEID,
        stm.LOVDELIVERYFAILUREREASONID = sd.LOVDELIVERYFAILUREREASONID,
        stm.CANCEL_DATE = sd.CANCEL_DATE,
        stm.CANCELEDBY = sd.CANCELEDBY,
        stm.CANCELED = sd.CANCELED,
        stm.AUDIT_CREATE_DATE = sd.AUDIT_CREATE_DATE,
        stm.AUDITCREATEUNITID = sd.AUDITCREATEUNITID,
        stm.AUDIT_MODIFY_DATE = sd.AUDIT_MODIFY_DATE,
        stm.AUDIT_DELETED = sd.AUDIT_DELETED,
        stm.PLANNEDDELIVERYDATEID = sd.PLANNEDDELIVERYDATEID,
        stm.ACTUALPLANNEDDELIVERYDATEID                                    =
        sd.ACTUALPLANNEDDELIVERYDATEID,
        stm.PLANNEDARRIVALDATEID = sd.PLANNEDARRIVALDATEID,
        stm.FIRSTPIECEARRIVALDATEID = sd.FIRSTPIECEARRIVALDATEID,
        stm.LASTPIECEARRIVALDATEID = sd.LASTPIECEARRIVALDATEID,
        stm.DELIVERYDATEID = sd.DELIVERYDATEID,
        stm.SHIPMENTDATEID = sd.SHIPMENTDATEID,
        stm.CANCELDATEID = sd.CANCELDATEID,
        stm.AUDITCREATEDATEID = sd.AUDITCREATEDATEID,
        stm.AUDITMODIFYDATEID = sd.AUDITMODIFYDATEID,
        stm.DUEDATEID = sd.DUEDATEID,
        stm.TRADINGDATEID = sd.TRADINGDATEID,
        stm.CARGO_CODE = sd.CARGO_CODE,
        stm.SENDERACCOUNTNO = sd.SENDERACCOUNTNO,
        stm.RECEIVERACCOUNTNO = sd.RECEIVERACCOUNTNO,
        stm.SENDERACCOUNTADDRESSVERSIONNO                                  =
        sd.SENDERACCOUNTADDRESSVERSIONNO,
        stm.INVOICEADDRESSVERSIONNO = sd.INVOICEADDRESSVERSIONNO,
        stm.INVOICEADDRESSNO = sd.INVOICEADDRESSNO,
        stm.RECEIVERADDRESSVERSIONNO = sd.RECEIVERADDRESSVERSIONNO,
        stm.WH_TRANSFER_DATE = sd.WH_TRANSFER_DATE,
        stm.AUDIT_MODIFY_DATE2 = sd.AUDIT_MODIFY_DATE2
         WHEN NOT MATCHED THEN INSERT (
              (OPERATION_DATE, SOURCEUNITID, SERIAL_NUMBER,SHIPMENT_CODE
             ,SENDERACCOUNTIDRECEIVERACCOUNTID,      WAYBILLIDSHIPMENTID,
CARGOID
             ,CAMPAIGNCONDITIONID, ACCOUNTCONTRACTVERSIONID, YI_YD
             ,PIECE_COUNT,                TOTAL_VOLUME,BIREYSEL_KURUMSAL,
PRIMARYSERVICEID
             ,TOTAL_PRICE      ,TOTAL_INVOICE_PRICE,      OPERATIONDATEID,
SATELLITEID
             ,PLANNED_DELIVERY_DATE, ACTUAL_PLANNED_DELIVERY_DATE
             ,PLANNED_ARRIVAL_DATE, FIRST_PIECE_ARRIVAL_DATE

,LAST_PIECE_ARRIVAL_DATE,DURATION,MOBILE,MANIFESTUNITID,DESCRIPTION
             ,INTEGRATION_CODE, CONTENTS_DESCRIPTION, DELIVERY_DATE
             ,SENDERACCOUNTNAME,
SENDERACCOUNTADDRESSNAME,RECEIVERACCOUNTNAME
```

```
            ,RECEIVERACCOUNTADDRESSNAME, INVOICEID, CANCELUNITID
            ,LOVWAYBILLTYPEID, CANCEL_DESCRIPTION, AUDITCREATEDBY
            ,AUDITMODIFIEDBYAUDITMODIFYUNITID,
SENDERACCOUNTADDRESSVERSIONID
            ,INVOICEUNITID, COLLECTIONUNITID, INVOICEADDRESSVERSIONID
            ,WORLDWIDE, LOVDOCUMENTPRINTSTATUSID, DIFFINVOICEID
            ,PAYORACCOUNTCUSTOMERID,       RETURNED,       PUANTUMBONUS,
PUANTUMCANCEL
            ,PUANTUMCANCELED,      PUANTUMCARDNO,      PUANTUMCARDOWNER,
PUANTUMWEB
            ,DUE_DATE, PAYORACCOUNTNAME, INVOICEADDRESSID
            ,ACCCONTRACTID, RECEIVERADDRESSVERSIONID, REFCODE
            ,TRADING_WAYBILL_NUMBER                        ,TRADING_GOODS
,RESPONSIBILITY_DOCUMENT
            ,RECEIVEEMPLOYEEID, MEASUREEMPLOYEEID, CONTACT_NAME
             ,LOVIDENTITYTYPEID,CONTACT_IDENTITY_OFFICE
             ,CONTACT_IDENTITY_NUMBER
            ,PAYMENTACCOUNTCONTRACTVERSID, PRICELISTID, CARGOCOLLECTID
            ,LOVCARGOSTATUSID, PARTY_CODE, LOVPARTNERID, CARGONOTICEID
            ,TRADING_DATE, FOR_WORLDWIDE, WWCARGO_VALUE, DELIVERYUNITID
            ,RESPONSIBLEUNITID, SHIPMENT_DATE, SHIPMENTDELIVEREDID
            ,PARENTDELIVERYUNITID, PARENTSOURCEUNITID, PRINTERBARCODEID
            ,LOVPAYORTYPEID, LOVPACKTYPEID, LOVSHIPMENTTYPEID
            ,LOVUNITDISTANCETYPEID,                LOVSHIPMENTSTATUSID,
LOVDELIVERYTYPEID
            ,LOVDELIVERYFAILUREREASONID,    CANCEL_DATE,    CANCELEDBY,
CANCELED
            ,AUDIT_CREATE_DATE, AUDITCREATEUNITID, AUDIT_MODIFY_DATE

,AUDIT_DELETED,PLANNEDDELIVERYDATEID,ACTUALPLANNEDDELIVERYDATEID
            ,PLANNEDARRIVALDATEID, FIRSTPIECEARRIVALDATEID
            ,LASTPIECEARRIVALDATEID, DELIVERYDATEID, SHIPMENTDATEID
            ,CANCELDATEID,     AUDITCREATEDATEID,     AUDITMODIFYDATEID,
DUEDATEID
            ,TRADINGDATEID,         CARGO_CODE,         SENDERACCOUNTNO,
RECEIVERACCOUNTNO
            ,SENDERACCOUNTADDRESSVERSIONNO, INVOICEADDRESSVERSIONNO
            ,INVOICEADDRESSNO, RECEIVERADDRESSVERSIONNO
            ,WH_TRANSFER_DATE, AUDIT_MODIFY_DATE2)
      VALUES (
      sd.OPERATION_DATE,
      sd.SOURCEUNITID,
      sd.SERIAL_NUMBER,
      sd.SHIPMENT_CODE,
      sd.SENDERACCOUNTID,
      sd.RECEIVERACCOUNTID,
      sd.WAYBILLID,
      sd.SHIPMENTID,
      sd.CARGOID,
      sd.CAMPAIGNCONDITIONID,
      sd.ACCOUNTCONTRACTVERSIONID,
      sd.YI_YD,
      sd.PIECE_COUNT,
      sd.TOTAL_VOLUME,
      sd.BIREYSEL_KURUMSAL,
      sd.PRIMARYSERVICEID,
      sd.TOTAL_PRICE,
      sd.TOTAL_INVOICE_PRICE,
      sd.OPERATIONDATEID,
```

```
sd.SATELLITEID,
sd.PLANNED_DELIVERY_DATE,
sd.ACTUAL_PLANNED_DELIVERY_DATE,
sd.PLANNED_ARRIVAL_DATE,
sd.FIRST_PIECE_ARRIVAL_DATE,
sd.LAST_PIECE_ARRIVAL_DATE,
sd.DURATION,
sd.MOBILE,
sd.MANIFESTUNITID,
sd.DESCRIPTION,
sd.INTEGRATION_CODE,
sd.CONTENTS_DESCRIPTION,
sd.DELIVERY_DATE,
sd.SENDERACCOUNTNAME,
sd.SENDERACCOUNTADDRESSNAME,
sd.RECEIVERACCOUNTNAME,
sd.RECEIVERACCOUNTADDRESSNAME,
sd.INVOICEID,
sd.CANCELUNITID,
sd.LOVWAYBILLTYPEID,
sd.CANCEL_DESCRIPTION,
sd.AUDITCREATEDBY,
sd.AUDITMODIFIEDBY,
sd.AUDITMODIFYUNITID,
sd.SENDERACCOUNTADDRESSVERSIONID,
sd.INVOICEUNITID,
sd.COLLECTIONUNITID,
sd.INVOICEADDRESSVERSIONID,
sd.WORLDWIDE,
sd.LOVDOCUMENTPRINTSTATUSID,
sd.DIFFINVOICEID,
sd.PAYORACCOUNTCUSTOMERID,
sd.RETURNED,
sd.PUANTUMBONUS,
sd.PUANTUMCANCEL,
sd.PUANTUMCANCELED,
sd.PUANTUMCARDNO,
sd.PUANTUMCARDOWNER,
sd.PUANTUMWEB,
sd.DUE_DATE,
sd.PAYORACCOUNTNAME,
sd.INVOICEADDRESSID,
sd.ACCCONTRACTID,
sd.RECEIVERADDRESSVERSIONID,
sd.REFCODE,
sd.TRADING_WAYBILL_NUMBER,
sd.TRADING_GOODS,
sd.RESPONSIBILITY_DOCUMENT,
sd.RECEIVEEMPLOYEEID,
sd.MEASUREEMPLOYEEID,
sd.CONTACT_NAME,
sd.LOVIDENTITYTYPEID,
sd.CONTACT_IDENTITY_OFFICE,
sd.CONTACT_IDENTITY_NUMBER,
sd.PAYMENTACCOUNTCONTRACTVERSID,
sd.PRICELISTID,
sd.CARGOCOLLECTID,
sd.LOVCARGOSTATUSID,
sd.PARTY_CODE,
```

```
        sd.LOVPARTNERID,
        sd.CARGONOTICEID,
        sd.TRADING_DATE,
        sd.FOR_WORLDWIDE,
        sd.WWCARGO_VALUE,
        sd.DELIVERYUNITID,
        sd.RESPONSIBLEUNITID,
        sd.SHIPMENT_DATE,
        sd.SHIPMENTDELIVEREDID,
        sd.PARENTDELIVERYUNITID,
        sd.PARENTSOURCEUNITID,
        sd.PRINTERBARCODEID,
        sd.LOVPAYORTYPEID,
        sd.LOVPACKTYPEID,
        sd.LOVSHIPMENTTYPEID,
        sd.LOVUNITDISTANCETYPEID,
        sd.LOVSHIPMENTSTATUSID,
        sd.LOVDELIVERYTYPEID,
        sd.LOVDELIVERYFAILUREREASONID,
        sd.CANCEL_DATE,
        sd.CANCELEDBY,
        sd.CANCELED,
        sd.AUDIT_CREATE_DATE,
        sd.AUDITCREATEUNITID,
        sd.AUDIT_MODIFY_DATE,
        sd.AUDIT_DELETED,
        sd.PLANNEDDELIVERYDATEID,
        sd.ACTUALPLANNEDDELIVERYDATEID,
        sd.PLANNEDARRIVALDATEID,
        sd.FIRSTPIECEARRIVALDATEID,
        sd.LASTPIECEARRIVALDATEID,
        sd.DELIVERYDATEID,
        sd.SHIPMENTDATEID,
        sd.CANCELDATEID,
        sd.AUDITCREATEDATEID,
        sd.AUDITMODIFYDATEID,
        sd.DUEDATEID,
        sd.TRADINGDATEID,
        sd.CARGO_CODE,
        sd.SENDERACCOUNTNO,
        sd.RECEIVERACCOUNTNO,
        sd.SENDERACCOUNTADDRESSVERSIONNO,
        sd.INVOICEADDRESSVERSIONNO,
        sd.INVOICEADDRESSNO,
        sd.RECEIVERADDRESSVERSIONNO,
        sd.WH_TRANSFER_DATE,
        sd.AUDIT_MODIFY_DATE2
        )option (merge join);


SET TRANSACTION ISOLATION LEVEL READ COMMITTED
exec ('UPDATE OPENQUERY (ESASLIVE, ''select * from whtransfer where
whtransferid='''''+@l_whtransferid+''''''')                          set
lovwhtransferstatusid=2,TRANSFER_COMPLETE_DATE=getdate()');

END

GO
```

APPENDIX C

Transact/SQL Source code of OLAP Cargo Cube

```
set transaction isolation level read uncommitted;


select    a11.LOVWAYBILLTYPEID  LOVWAYBILLTYPEID,
          a11.DELIVERYUNITID  DELIVERYUNITID,
          a11.SHIPMENTDATEID  SHIPMENTDATEID,
          a11.SOURCEUNITID  SOURCEUNITID,
          a11.WORLDWIDE  WORLDWIDE,
          a11.SATELLITEID  SATELLITEID,
          a11.OPERATIONDATEID  OPERATIONDATEID,
          a11.PLANNEDARRIVALDATEID  PLANNEDARRIVALDATEID,
          a11.PLANNEDDELIVERYDATEID  PLANNEDDELIVERYDATEID,
          a11.AUDITCREATEDATEID  AUDITCREATEDATEID,
          a11.LOVUNITDISTANCETYPEID  LOVUNITDISTANCETYPEID,
          a11.LOVPACKTYPEID  LOVPACKTYPEID,
          a11.LOVSHIPMENTSTATUSID  LOVSHIPMENTSTATUSID,
          a11.campaign_f  campaign_f,
          a11.ACTUALPLANNEDDELIVERYDATEID  ACTUALPLANNEDDELIVERYDATEID,
          a11.LOVSHIPMENTTYPEID  LOVSHIPMENTTYPEID,
          a11.SERIAL_NUMBER  SERIAL_NUMBER,
          a11.BIREYSEL_KURUMSAL  BIREYSEL_KURUMSAL,
          a11.contract_f  contract_f,
          a11.LOVMAINPACKTYPEID  LOVPACKTYPEID0,
          a11.campaignconditionid  campaignconditionid,
          a11.PRIMARYSERVICEID  SERVICEID,
          sum((Case  when  a11.TOTAL_INVOICE_PRICE  =  0  then  0  else
a11.TOTAL_INVOICE_PRICE end))  WJXBFS1,
          sum(a11.TOTAL_VOLUME)  WJXBFS2,
          sum(a11.PIECE_COUNT)  WJXBFS3,
          count(a11.WAYBILLID)  WJXBFS4
into #ZZMD00
from      dbo.CARGO      a11
          join      OPERATIONDATE      a12
            on      (a11.OPERATIONDATEID = a12.OPERATIONDATEID)
where     (a12.OPERATION_DATE >=  '2011-07-01'
 and a12.OPERATION_DATE < '2011-08-22')
group by          a11.LOVWAYBILLTYPEID,
          a11.DELIVERYUNITID,
          a11.SHIPMENTDATEID,
          a11.SOURCEUNITID,
          a11.WORLDWIDE,
          a11.SATELLITEID,
          a11.OPERATIONDATEID,
          a11.PLANNEDARRIVALDATEID,
          a11.PLANNEDDELIVERYDATEID,
          a11.AUDITCREATEDATEID,
          a11.LOVUNITDISTANCETYPEID,
          a11.LOVPACKTYPEID,
          a11.LOVSHIPMENTSTATUSID,
```

```
        a11.campaign_f,
        a11.ACTUALPLANNEDDELIVERYDATEID,
        a11.LOVSHIPMENTTYPEID,
        a11.SERIAL_NUMBER,
        a11.BIREYSEL_KURUMSAL,
        a11.contract_f,
        a11.LOVMAINPACKTYPEID,
        a11.campaignconditionid,
        a11.PRIMARYSERVICEID

select  distinct pa13.WORLDWIDE  WORLDWIDE,
        a122.NAME  NAME,
        pa13.SERIAL_NUMBER  SERIAL_NUMBER,
        pa13.SHIPMENTDATEID  SHIPMENTDATEID,
        a123.shipmentdate  shipmentdate,
        pa13.LOVSHIPMENTTYPEID  LOVSHIPMENTTYPEID,
        a112.NAME  NAME0,
        pa13.LOVSHIPMENTSTATUSID  LOVSHIPMENTSTATUSID,
        a115.NAME  NAME1,
        pa13.LOVWAYBILLTYPEID  LOVWAYBILLTYPEID,
        a124.NAME  NAME2,
        pa13.LOVPACKTYPEID  LOVPACKTYPEID,
        a116.NAME  NAME3,
        a16.PARENTUNITID  PARENTSOURCEUNITID,
        SUBSTRING(a126.NAME, 1, (LEN(a126.NAME) - 6))  CustCol_6,
        pa13.SOURCEUNITID  SOURCEUNITID,
        a16.NAME  NAME5,
        pa13.LOVUNITDISTANCETYPEID  LOVUNITDISTANCETYPEID,
        a117.NAME  NAME6,
        pa13.contract_f  contract_f,
        a111.NAME  NAME7,
        pa13.campaign_f  campaign_f,
        a114.NAME  NAME8,
        pa13.SERVICEID  SERVICEID,
        a18.NAME  NAME9,
        pa13.ACTUALPLANNEDDELIVERYDATEID  ACTUALPLANNEDDELIVERYDATEID,
        a113.actualplanneddeliverydate  actualplanneddeliverydate,
        pa13.AUDITCREATEDATEID  AUDITCREATEDATEID,
        a118.auditcreatedate  auditcreatedate,
        pa13.PLANNEDDELIVERYDATEID  PLANNEDDELIVERYDATEID,
        a119.planneddeliverydate  planneddeliverydate,
        pa13.PLANNEDARRIVALDATEID  PLANNEDARRIVALDATEID,
        a120.plannedarrivaldate  plannedarrivaldate,
        pa13.OPERATIONDATEID  OPERATIONDATEID,
        a14.OPERATION_DATE  OPERATION_DATE,
        pa13.LOVPACKTYPEID0  LOVPACKTYPEID0,
        a110.NAME  NAME10,
        a14.DAYOFWEEKID  DAYOFWEEKID,
        a15.NAME  NAME11,
        a15.WEEKDAY  WEEKDAY,
        a15.WEEKDAY  WEEKDAY0,
        pa13.BIREYSEL_KURUMSAL  BIREYSEL_KURUMSAL,
        pa13.campaignconditionid  campaignconditionid,
        a19.campaign_name  campaign_name,
        pa13.DELIVERYUNITID  DELIVERYUNITID,
        a17.NAME  NAME12,
        a16.SENDERHUBUNITID  SENDERHUBUNITID,
        a127.NAME  NAME13,
```

```
        pa13.SATELLITEID  SATELLITEID,
        a121.NAME  NAME14,
        a16.hintcityid  hintcityid,
        a125.CITY  CITY,
        a17.hintcityid  hintcityid0,
        a128.CITY  CITY0,
        pa13.WJXBFS1  WJXBFS1,
        pa13.WJXBFS2  WJXBFS2,
        pa13.WJXBFS3  WJXBFS3,
        pa13.WJXBFS4  WJXBFS4
from    #ZZMD00         pa13
        join    OPERATIONDATE       a14
          on    (pa13.OPERATIONDATEID = a14.OPERATIONDATEID)
        join    DAYOFWEEK   a15
          on    (a14.DAYOFWEEKID = a15.DAYOFWEEKID)
        join    V_SOURCEUNIT        a16
          on    (pa13.SOURCEUNITID = a16.SOURCEUNITID)
        join    dbo.v_firstdeliveryunit     a17
          on    (pa13.DELIVERYUNITID = a17.DELIVERYUNITID)
        join    V_PRIMARYSERVICES a18
          on    (pa13.SERVICEID = a18.PRIMARYSERVICEID)
        join    V_CAMPAIGNCONDITION       a19
          on    (pa13.campaignconditionid = a19.campaignconditionid)
        join    dbo.LOVMAINPACKTYPE       a110
          on    (pa13.LOVPACKTYPEID0 = a110.LOVMAINPACKTYPEID)
        join    V_CONTRACT_F        a111
          on    (pa13.contract_f = a111.contract_f)
        join    LOVSHIPMENTTYPE     a112
          on    (pa13.LOVSHIPMENTTYPEID = a112.LOVSHIPMENTTYPEID)
        join    V_ACTUALPLANNEDDELIVERYDATE a113
          on    (pa13.ACTUALPLANNEDDELIVERYDATEID                                   =
a113.ACTUALPLANNEDDELIVERYDATEID)
        join    V_CAMPAIGN_F        a114
          on    (pa13.campaign_f = a114.campaign_f)
        join    LOVSHIPMENTSTATUS       a115
          on    (pa13.LOVSHIPMENTSTATUSID = a115.LOVSHIPMENTSTATUSID)
        join    LOVPACKTYPE         a116
          on    (pa13.LOVPACKTYPEID = a116.LOVPACKTYPEID)
        join    LOVUNITDISTANCETYPE       a117
          on    (pa13.LOVUNITDISTANCETYPEID = a117.LOVUNITDISTANCETYPEID)
        join    V_AUDITCREATEDATE       a118
          on    (pa13.AUDITCREATEDATEID = a118.AUDITCREATEDATEID)
        join    V_PLANNEDDELIVERYDATE  a119
          on    (pa13.PLANNEDDELIVERYDATEID = a119.PLANNEDDELIVERYDATEID)
        join    V_PLANNEDARRIVALDATE    a120
          on    (pa13.PLANNEDARRIVALDATEID = a120.PLANNEDARRIVALDATEID)
        join    dbo.V_SATELLITE       a121
          on    (pa13.SATELLITEID = a121.SATELLITEID)
        join    V_WORLDWIDE         a122
          on    (pa13.WORLDWIDE = a122.WORLDWIDE)
        join    V_SHIPMENTDATE      a123
          on    (pa13.SHIPMENTDATEID = a123.SHIPMENTDATEID)
        join    LOVWAYBILLTYPE      a124
          on    (pa13.LOVWAYBILLTYPEID = a124.LOVWAYBILLTYPEID)
        join    dbo.v_sourceunitcity        a125
          on    (a16.hintcityid = a125.hintcityid)
        join    V_PARENTSOURCEUNIT        a126
          on    (a16.PARENTUNITID = a126.SOURCEUNITID and
```

```
        a16.lovorganizationid = a126.lovorganizationid)
        join    dbo.V_SENDERHUBUNIT         a127
          on    (a16.SENDERHUBUNITID = a127.SENDERHUBID)
        join    dbo.v_firstdeliveryunitcity         a128
          on    (a17.hintcityid = a128.hintcityid)


drop table #ZZMD00
```

CURRICULUM VITAE

Name Surname : Mert Sun

Address : Barbaros Bulvarı, Ressam Hamdi Bey Sok. No:1, Müge Apt. D:14, Beşiktaş-İSTANBUL

Birth Place and Date: İzmir, 22.10.1972

Foreign Language : English

Education : Bahçeşehir University, 2011
M.S. Computer Engineering

İstanbul Teknik University, 1999
Metalurgy and Material Engineering

İzmir Anatolian Commercial High School, 1990

Work Experience : Aras Kargo Taşımacılık A.Ş. (2008 -)
Software Manager Deputy

Aras Kargo Taşımacılık A.Ş. (2002 – 2008)
Senior Software Engineer