WEB SERVICES BASED REAL TIME DATA WAREHOUSE

MURAT OBALI

JULY, 2012

Title of the Thesis : **Web Services Based Real Time Data Warehouse**

Submitted by **Murat Obalı**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Murat SARAN
Head of Department

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

Dr. Zeki ERDEM
Co-Supervisor

Assist. Prof. Dr. Abdül Kadir GÖRÜR
Supervisor

Examination Date : **18.07.2012**

Examining Committee Members:

Assist. Prof. Dr. Abdül Kadir GÖRÜR (Çankaya Univ.)

Assist. Prof. Dr. Reza ZARE HASSANPOUR (Çankaya Univ.)

Dr. Ersin ELBAŞI (TÜBİTAK)

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also, declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Murat Obalı

Signature : 

Date : 18.07.2012

ABSTRACT

WEB SERVICES BASED REAL TIME DATA WAREHOUSE

Obalı, Murat

M.S.c., Department of Computer Enginnering

Supervisor: Assist. Prof. Dr. Abdül Kadir GÖRÜR

Co-Supervisor: Dr. Zeki ERDEM

July 2012, 84 pages

Today's business environment is quickly changing and business decision makers need for a historical picture of what happened and a picture of what was happening today. Traditional data warehouses provide a historical picture, but there is lack of fresh data. However, fresh data in data warehouses is a strong feature from the part of the users. The aim of this study is building a real time data warehouse using web services. First, we modelled both the conceptual and the logical design of real time data warehouse. For change data capture from source systems, we implemented web services based server and client software. Then, we used real time partition for real time data which is merged into data warehouse in a daily fashion. We, also, implemented a data integration service using query re-write approach to integrate data warehouse and real time partition data.

Keywords: Real Time Data Warehouse, Data Warehouse, Web Service, Real Time Partition, Clean Delta, On Demand Aggregation

ÖZ

WEB SERVİSLERİ TABANLI GERÇEK ZAMANLI VERİ AMBARI

Obalı, Murat

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi: Yrd. Doç. Dr. Abdül Kadir GÖRÜR

Ortak Tez Yöneticisi: Dr. Zeki ERDEM

Temmuz 2012, 84 pages

Günümüz iş dünyası çok hızlı değişmektedir ve karar vericilerin geçmişte neler olduğuna ve bugün neler olmakta olduğuna dair bir resme ihtiyaçları vardır. Geleneksel veri ambarları tarihsel resmi sağlamaktadır, fakat taze veriden yoksunlardır. Oysa ki, veri ambarlarındaki taze veri kullanıcı açısından oldukça önemli bir özelliktir. Bu çalışmanın amacı web servisleri kullanarak gerçek zamanlı bir veri ambarı geliştirmektir. İlk olarak, gerçek zamanlı veri ambarının konsept ve mantıksal modellemesini yaptık. Kaynak sistemlerdeki değişen verileri yakalamak için web servis tabanlı istemci-sunucu yazılımı geliştirdik. Daha sonra, veri ambarına günlük bazda yükleyeceğimiz veriler için gerçek zamanlı bölüm oluşturduk. Ayrıca, sorgu yeniden yazma yaklaşımını kullanarak, veri ambarı ile gerçek zamanlı bölüm verilerini birleştirmek için bir veri entegrasyon servisi gerçekleştirdik.

Keywords: Gerçek Zamanlı Veri Ambarı, Veri Ambarı, Web Servisi, Gerçek Zamanlı Bölüm, Temiz Fark, Taleb Bazlı Toplama

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my supervisor Assist. Prof. Dr. Abdül Kadir GÖRÜR and co-supervisor Dr. Zeki ERDEM for their guidance, advices, criticism, encouragements, and insight throughout the research.

I should also express my appreciation to examination committee members for their valuable suggestions and comments.

I would like to express my thanks to Fırat KÜÇÜK for his suggestions and comments.

I would like to express my thanks to my wife for her assistance, encouragement and all members of my family for their patience, sympaty and support during the study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES **PAGES**

LIST OF FIGURES

FIGURES **PAGES**

CHAPTER 1

INTRODUCTION

Today's business environment is quickly changing and business decision makers need for a historical picture of what happened and a picture of what was happening today. Engineers strongly defended the notion that the data warehouse needed to provide a reliable information floor upon which to stand, providing an unwavering set of data to business decision makers. Because of the twinkling database, business users were directed to the production applications that run the business for up-to-the-moment reporting. Therefore, users had to go to the data warehouse for a historical picture of what happened in the business as of yesterday and had to look across many OLTP systems for a picture of what was happening today. This division never fully accepted by business users. They want to go to one place to get the business information that they needed [19].

Fresh data in data warehouses is a strong feature from the part of the users. Traditionally, loading data into warehouses has been performed in an off-line period. In such a data warehouse setting, data are extracted from the sources, transformed, cleaned, and loaded to the warehouse. To avoid overloading the source production systems, data warehouse activities takes place during a loading window, usually during the night. In most cases, a data warehouse is typically updated every day (24 hours period) [39].

The delay between a business transaction and its appearance in the data warehouse is too much for many organizations in fast-moving vertical industries. Additionally the data warehouse has become mission critical. That is, feeding enriched information back to operational systems that is then used to process transactions; personalize offers, and present up-sell promotions. The push for ever-fresher information is needed [19].

The traditional data warehouses are implemented using batch driven approach and mainly according to the pull technology principle. The data loading from source systems to data warehouses is generally performed on a nightly basis or even in some cases on a weekly basis; therefore typical data warehouses normally do not have the most current data [39]. Furthermore the operational systems may have to be go offline during the data extraction process. It is an unacceptable situation which generates delays in businesses especially that require instantaneous access to up-to-date information [44].

A real time data warehouse eliminates the data availability gap and enables organizations to concentrate on processing their valuable data. Furthermore, continuous data processing without delay opens up significant new opportunities [46].

The zero-latency enterprise is ideal for a business. This ideal urges the benefits of speed and a single version of the truth. In a real time, generally mean, information is delivered to the right place at the right time for maximum business value. We may call these *right-time* systems. At present, true zero latency is an unattainable ideal—it takes some time to synchronize information across several production systems and data warehouse—but there is a pressure on many modern data warehouses to provide a low-latency view of the business [19].

In summary, we may list three major reasons why we need a Real Time in a Data Warehouse:

- Faster reaction time
- Reduced decision time
- New process capabilities.

CHAPTER 2

DATABASE

2.1 Definition of Database

Databases and database technology are an essential component and play a critical role in almost all areas where computers are used, including business, engineering, medicine, law, education, and library science, to name a few. The growing use of computers increases its impact. Database technology is the corner stone of most of the modern information systems. Therefore, it will be good to begin with the word database.

"*A **database** is a collection of related data. By **data**, we mean known facts that can be recorded and that have implicit meaning*" [1]. For example, consider the list products that a company manufactured or names and telephone numbers of customers. This data may have been recorded in an indexed address book, or may have been stored on a hard disk or flash memory, using a personal computer and software such as Microsoft ACCESS or EXCEL. The collection of related data usually referred to as the database.

In addition, the common use of the term database is usually more restricted. With own words of Ramez and Navathe, a database has the following implicit properties:

- "*A database represents some aspect of the real world, sometimes called the **miniworld** or the **universe of discourse (UoD).** Changes to the miniworld are reflected in the database*".
- "*A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database*".

- "*A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested*".

So to summarize, a database has some source from which data is taken, a degree of interaction with the events in the real world, and an audience that is actively interested in the contents of a database [1].

The size and complexity of a database can vary. The Çankaya University library database may include hundreds of thousands of items, whereas a small company may maintain a database for only 20 employees.

2.1.1 Database Management System

The knowledge and technology that has developed over several decades resulted in specialized software called a **database management system (DBMS)** or generally a "**database system**". With a DBMS, you can create and manage large amounts of data efficiently. Also it is a powerful tool for allowing it to persist over long periods of time, safely. These systems are among the most complex types of software available [2]. Some of the world's most popular databases are: Oracle, DB2, Microsoft SQL Server and MySQL.

2.1.2 Data Model

The main purpose of the database systems is to manage large bodies of information. This data management involves both defining structures for storage of information and providing mechanisms for the manipulation of information. Also, the safety of the information stored must be ensured by the database system, despite system crashes or attempts at unauthorized access. If several users have to share data, the system must avoid possible anomalous results. Because information is so important in most organizations, a large body of concepts and techniques for managing data has been developed [3].

Database systems can be based on different data models or database models respectively. Data modeling is a way for specifying the structures of data. A data model is a collection of concepts and rules for the description of the structure of the database. Data types, the constraints and the relationships for the description or storage of data are main structures of the database. The most often used data models are [4]:

Network Model and Hierarchical Model	The network model and the hierarchical model build upon individual data sets and are able to express hierarchical or network like structures of the real world. They are the predecessors of the relational model.
Relational Model	The relational model defines a database as a collection of tables (relations) which contain all data. It is the best known and in today's DBMS most often implemented database model.
Object-Oriented	Object-oriented models define a database as a collection of objects with features and methods.
Object-Relational Model	Object-relational database model is the wide spread and simple relational database model extended by some basic object-oriented concepts. Object-oriented models are very powerful but also quite complex, therefore object relational model allow us to work with the widely know relational database model but also have some advantages of the object-oriented model without its complexity.

The **relational model** is today the primary data model for commercial data processing applications. It attained its primary position because of its simplicity, which eases the job of the programmer, compared to earlier data models such as the network model or the hierarchical model. Relational model is developed by E.F. Codd and a database based on this model allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized in tables which are a collection of records [3].

2.1.3 Transaction

A **transaction** is a unit of work submitted to a database by a single database user or collections of operations that form a single logical unit of work. Transactions are important to multiuser databases because databases provide many concurrency control mechanisms by using transactions that is either

succeed or fail as a whole. A database system must ensure proper execution of transactions despite failures. Also, it must manage concurrent execution of transactions in a way that avoids the inconsistency [3] [5].

In a transaction various data items are accessed and possibly updated or deleted. To initiate a transaction, usually a user program is used, which is written in a high-level data-manipulation language (typically SQL), or programming language (for example, C++, or Java), with embedded database accesses in JDBC or ODBC. A transaction is delimited by statements of the form "begin **transaction**" and "**end transaction**" and the transaction consists of all operations executed between the begin transaction and end transaction [3].

2.1.4 OLTP vs. OLAP

Database systems can be divided into transactional (OLTP) and analytical (OLAP). In general we can say that OLTP systems provide source data to data warehouses, whereas OLAP systems help to analyze it [6] [7].

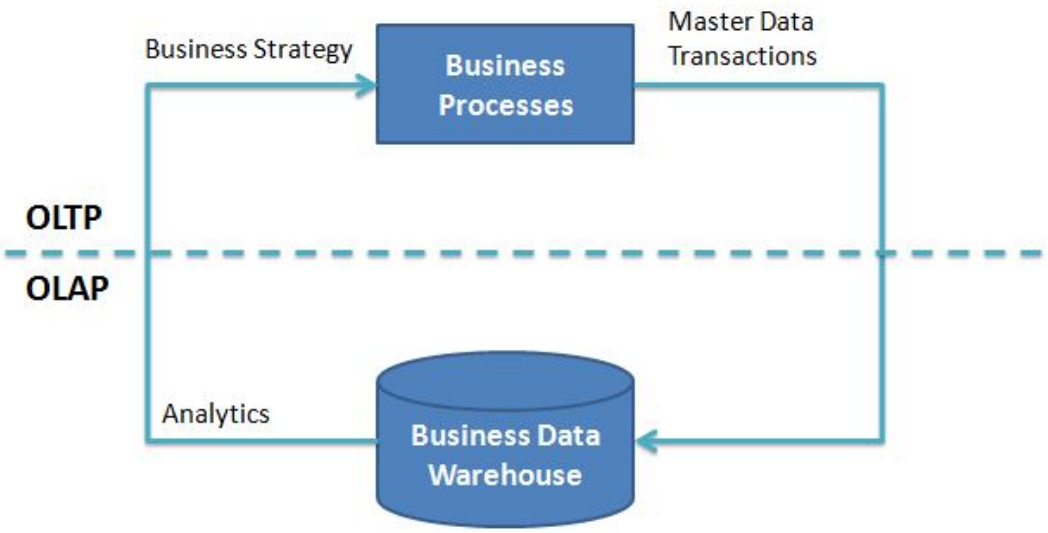

Figure 1: OLTP and OLAP Systems [6]

An **OLTP (On-line Transaction Processing)** deal with operational data and it is characterized by a large number of short on-line transactions (INSERT, UPDATE, and DELETE). The main importance for OLTP systems is provide very fast query processing, maintaining data integrity in multi-access environments. It's effectiveness generally measured by number of transactions per second. There is detailed and current data in OLTP database, and the entity model schema is used to store transactional databases. Additionally, the data is frequently updated and

queried in an OLTP system. The database tables are normalized to prevent data redundancy and to prevent update anomalies. This makes the write operation in the database tables more efficient [6] [7].

An **OLAP (On-line Analytical Processing)** is deal with Historical Data or Archival Data, and it is characterized by relatively low volume of transactions. Queries are often very complex and involve aggregations. Response time is an effectiveness measure for OLAP systems. OLAP applications are widely used by Data Mining techniques. There is aggregated, historical data, stored in multi-dimensional schemas in OLAP database [6] [7].

The design of a data warehouse database and online analytical processing (cubes, star schema etc) is fundamentally different than a transactional processing database. The data warehouse is particularly designed to facilitate super fast query times in a large dataset and multi-dimensional analysis. The following table summarizes the major differences between OLTP and OLAP system design [8].

Table 1: The major differences between OLTP and OLAP system design [8]

	OLTP System Online Transaction Processing (Operational System)	**OLAP System Online Analytical Processing (Data Warehouse)**
Source of data	Operational data; OLTPs are the original source of the data.	Consolidation data; OLAP data comes from the various OLTP Databases
Purpose of data	To control and run fundamental business tasks	To help with planning, problem solving, and decision support
What the data	Reveals a snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and Updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries Returning relatively few records	Often complex queries involving aggregations
Processing Speed	Typically very fast	Depends on the amount of data involved; batch data refreshes and complex queries may take many hours; query speed can be improved by creating indexes
Space Requirements	Can be relatively small if historical data is archived	Larger due to the existence of aggregation structures and

		history data; requires more indexes than OLTP
Database Design	Highly normalized with many tables	Typically de-normalized with fewer tables; use of star and/or snowflake schemas
Backup and Recovery	Backup religiously; operational data is critical to run the business, data loss is likely to entail significant monetary loss and legal liability	Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method

CHAPTER 3

DATA WAREHOUSE

3.1 Definition of Data Warehouse

The term **data warehouse (DW)** is commonly used in industry and it denote to a kind of heterogeneous information system. We have to disclose firstly that a data warehouse is an environment, not a product. The need for building a data warehouse is that corporate data is often scattered in different databases and possibly in different formats. In order to view a complete picture of information, it is necessary to access these heterogeneous databases. Therefore we have to obtain data and pieces of partial information from each, and then put them together to produce an overall picture. Attempting this process without a data warehouse is a cumbersome task, inefficient, ineffective, error-prone. Moreover this task usually will need huge efforts of system analysts. All these difficulties discourage the effective use of complex, but valuable corporate data [9].

The definition of data warehouse has evolved since its origins in the early 1980s. Some of the more common definitions [10]:

- Data Warehouse is a repository of subject-oriented, historical data.
- Data Warehouse is a collection of smaller “data marts”.
- Data Warehouse can be considered any separate hardware platform that enables a business person to make a decision.

All of these definitions can be correct, depending on your environment. Dyche [10] defines the data warehouse as a separate platform – a computer different from other computers in your IT environment.

A data warehouse consolidates and integrates information from many internal and external sources and arranges it in a meaningful format for making accurate and timely business decisions. In fact, data warehouse is a database and mainly used for reporting and analysis purposes. It is different from the organization's Online Transaction Processing (OLTP) database. The data stored in the warehouse is uploaded from the operational systems.

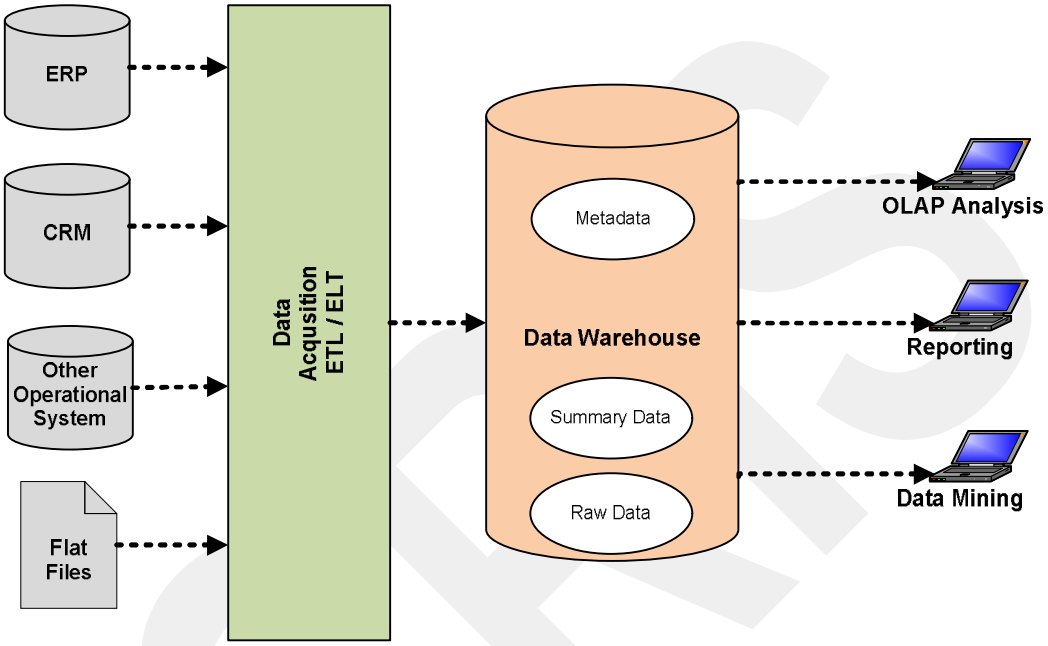

Figure 2: Sample Data Warehouse Architecture [11]

Bill Inmon [11] defines the term DW as: "*A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management's decisions*". This definition contains four key elements that are worthy of a detailed explanation:

- **Subject-Oriented:** In the data warehouse, all the data elements relating to the same real-world event are organized or object are linked together. The data in the warehouse should be organized based on subject, that is only subject-oriented data should be moved into a warehouse; for example, it can be product sales focusing on client interests in some sales company, the client behavior in utilization of different banking services, the insurance history of the clients, the railroad system utilization or changes in structure, etc.

- **Integrated:** DW is an architecture constructed by integrating data from multiple heterogeneous sources (such as relational database (DB), flat

files, excel sheets, XML data, data from the legacy systems) to support structured and/or ad hoc queries, analytical reporting and decision making. In this process, some problems have to be resolved: differences in data format, data codification, synonyms (fields with different names but the same data), homonyms (fields with the same name but different meaning), multiplicity of data occurrences, nulls presence, default values selection, etc.

- **Non-Volatile:** The data in the data warehouse can neither be modified nor removed, that is durable. Once committed, data in the data warehouse are never over-written or deleted. The data are static, read-only, and retained for future reporting.
- **Time-Variant:** DW is time variant in the sense that they maintain both historical and (nearly) current data. In contrast, operational databases contain only the most current, current (up-to-date) data values. DW provides information from a historical prospective. Therefore, every key structure in the DW contains, either implicitly or explicitly, an element of time. This indicates the possibility to count on different values of the same object according to its changes in time. For example, in a banking DW, the average balances of client's account during different months for the period of several years.

On the other hand, Ralph Kimball [12] briefly defines a DW as "*a copy of transaction data specifically structured for query and analysis*". He provides a more precise definition by means of requirements:

1. The data warehouse provides access to corporate or organizational data.
2. The data in a data warehouse is consistent.
3. The data in a data warehouse can be separated and combined by means of every possible measure in a business (the classic slice and dice requirement).
4. The data warehouse is not just data, but also a set of tools to query, analyze, and present information.
5. The data warehouse is the place where we publish used data.
6. The quality of the data in the data warehouse is a driver of business reengineering

Alternatively, Han and Kamber [13], define a DW as "*a repository of multiple heterogeneous data sources organized under a unified schema at a single site to facilitate management decision making*".

Finally, other authors focus their interest on the final users of the DW. For example, in [14], a DW is defined as a "*collection of technologies aimed at enabling the knowledge worker (executive, manager, and analyst) to make better and faster decisions*".

After these definitions we can focus on the four general principles of data warehousing listed below. These are true regardless of the platform, amount of data, and software being used [10].

1. A data warehouse is usually a separate computer, or hardware platform. This platform may be large or small and in some cases it might a collection of distributed platforms. In other words, it could be a set of "nodes" on a large computer platform.
2. The data on the data warehouse is used for decision making.
3. Data warehouses duplicate data that already exists elsewhere in the business. While this data redundancy sounds wasteful, it's actually a very good thing.
4. A data warehouse is not just a computer sitting someplace in the bowels of your companies data center. It's a combination of hardware, specialized software, and data. Normally, when people refer to "our data warehouse", they are talking about a hardware box, a collection of software products and tools, and lots and lots of data.

In short, a data warehouse is a repository of information extracted from other corporate systems such as transactional systems, departmental databases, company's intranet or Internet.

3.2 Data Warehouse, Decision Support, and Business Intelligence

Even before adopting data warehousing, an executive's assistant would be requesting a report for a certain product or some success rates from the IT department. While the IT person was querying the data and preparing the report, executive was waiting along day(s)/week(s). Figure 3 illustrates a typical lifecycle for acquiring valuable business information [10].

Figure 3: The Historical Process of Retrieving Information [10]

It is important here to understand the difference between data and information. The data warehouse synthesizes some very important data, but only when this data is combined into meaningful answers or reports that can support the interpretation of business events, then it can be considered information. While data has been difficult enough for companies to find and process, information has been next to impossible to obtain. Data in a data warehouse is cleansed and consolidated for access by variety of purposes [10].

One of the most important human activities is decision-making. It is more difficult in today's complex and rapidly changing decision environment than ever before. Decision Support Systems (DSS) are playing an important role in organizational decision-making for business intelligence in all disciplines, including health, business, engineering, education and finance. Organizational decision makers' requirements are increasing for advanced knowledge, previous successful experiences, and intelligent technical conditions to support and enable better decisions. In current advanced DSS, computational intelligence and knowledge-based methods and new analytical intelligence techniques, have become essential components. The ever-increasing distributed decision situations and related computing systems have triggered the development of a new generation of Intelligent Decision Support Systems (IDSS) [16].

In the new digital economy, rapid, relentless change is the only constant. So that companies must be able to forecast and adapt to ever-evolving market conditions to compete such a change. The key to achieving rapid and fast strategic performance is maintaining a steady flow of fully-integrated, actionable information about all key business areas, including production, customer service, supply, marketing, sales, and HR [17].

However, Liautaud [17] says that, when it comes to corporate intelligence, most companies are still plodding along at the speed of the steam-driven locomotive.

3.3 Definitions of Some Data Warehouse Terms

Term	Definition
Aggregation	A summarized, typically additive value. The level of aggregation depends on the scenario. Many star schemas are aggregated to some base level, called the grain, although this is becoming somewhat less common as developers rely on cube building engines to summarize to a base level of granularity [18].
Change Data Capture	Change Data Capture (CDC) is a generic term for techniques that monitor operational data sources with the objective of detecting and capturing data changes of interest [19].
Drill-down	The process of probing beyond a summarized value to investigate each of the detail transactions that comprise the summary.
Drill-up	A way of viewing related items of a Dimension as defined in the Hierarchy by collapsing members to come up to a summarized data range, or simply put, to hide child members associated with a specific parent or aggregate member within a defined hierarchy [20].
ETL / ELT	The Extract-Transform-Load (ETL) or Extract-Load-Transform system is the foundation of any data warehouse [19]. The objective of the ETL/ELT system is extracting data from multiple, heterogeneous data sources, transforming and cleansing data, and finally loading data into the data warehouse where it is accessible to business intelligence applications [21].
Grain	A definition of the highest level of detail that is supported in a data warehouse.
Incremental Load	“Full reloading is obviously inefficient considering that most often only a small fraction of source data is changed during loading cycles. It is rather desirable to capture source data changes and propagate the mere changes to the data warehouse. This approach is known as incremental loading” [21].
Initial Load	“The first population of a data warehouse is referred to as *initial load*. During an initial load, data is typically extracted exhaustively from the sources and delivered to the data warehouse. As source data changes over time, the data

	warehouse gets stale, and hence, needs to be refreshed. Data warehouse refreshment is typically performed in batch mode on a periodical basis. The naive approach to data warehouse refreshment is referred to as *full reloading*. The idea is to simply rerun the initial load job, collect the resulting data, and compare it to the data warehouse content. In this way, the required changes for data warehouse refreshment can be retrieved. Note that it is impractical to drop and recreate the data warehouse since historic data has to be maintained" [21].
Pivot Table	A pivot table is a program tool that allows you to reorganize and summarize selected columns and rows of data in a spreadsheet or database table to obtain a desired report. A pivot table doesn't actually change the spreadsheet or database itself. In database lingo, to pivot is to turn the data (slice and dice) to view it from different perspectives [22].
Rollup	Relational Online Analytical Processing. ROLAP is a flexible architecture that scales to meet the widest variety of DSS and OLAP needs. ROLAP architecture access data directly from data warehousing using SQL [23].

3.4 Major Components of Data Warehousing

The main reason for building a DW is to clean, consolidate and integrate the data coming from different sources, and therefore to improve the quality of information for making accurate and timely business decisions. We can list the major components of data warehousing as follows:

1. Data Sources
2. Data Extraction / Data Acquisition
3. Change Data Capture
4. Data Transformation
5. Data Loading
6. Data Staging
7. Operational Data Store (ODS)
8. Data Integration
9. Comprehensive Database
10. Metadata
11. Middleware Tools (enable access to the DW)

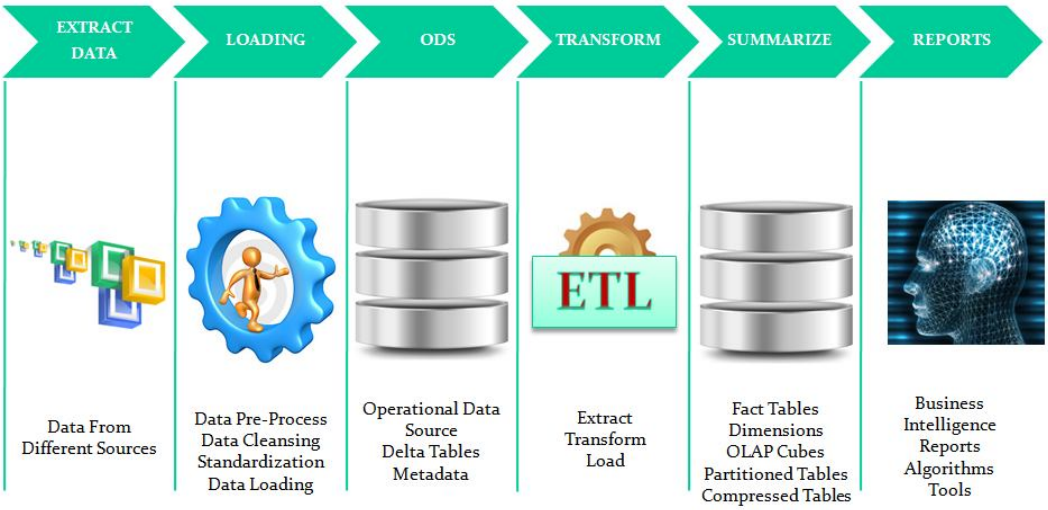

Figure 4: Major Data Warehouse Components

In Figure 4, major components of a data warehouse are shown in their architectural places.

3.4.1 Data Sources

Data Source, as the name intimates, provides data to DW via data site. Data site successively stores an organization's database, data files including non-automated data. A data source can be a relational database or a non-relational data source.

Data sources have to be identified before starting to develop data warehouse. The very first step needed to figure out what are the data that are required to be put into your data warehouse [24].

3.4.2 Data Extraction / Data Acquisition

Data extraction is the process of retrieving data from data sources for further data processing or data storage and it is a very important element of data warehouse implementation.

Extracting data from operational systems, and transform it into a format suitable for applications that will run off the data warehouse is an important part of the data warehouse implementation [9].

3.4.3 Change Data Capture

Capturing changes of data is crucial in refreshing data warehouse. Refreshing process begins with transferring the latest source data into the data warehouse.

However, we must transfer only the relevant changes to the source data since the last transfer because of the difficulty of complete refreshing. Completely refreshing our target fact and dimension tables are usually undesirable [25].

Isolating the latest source data is called **change data capture** and is often abbreviated CDC in high level architecture diagrams. The idea behind change data capture is to transfer the data that has been changed since the last load, but building a good change data capture system is not as easy. Some of the goals we have for capturing changed data [24]:

- Rather than complete refresh, use selective processing to isolate the changed source data
- Capture all source data changes (deletions, edits and insertions) including changes made through non-standard interfaces
- To distinguish error corrections from true updates, label changed data with reason codes
- Use additional metadata to support compliance tracking
- Start the change data capture process as early as possible, preferably before bulk data transfer to data warehouse

Detecting the changes is the first step in change data capture and there are mainly four ways to detect changes [25] [26]:

- **Audit columns:** The source system contains audit columns that stores the date and time a record was added or modified.
- **Database log scraping or sniffing:** Log scraping effectively takes a snapshot of the database redo log at a scheduled point in time (usually midnight) and scours it for transactions that affect the tables for ETL load. Sniffing involves a “polling” of the redo log, capturing transactions on-the-fly.
- **Timed extracts:** With a timed extract you typically select all of the rows where the date in the Create or Modified date fields equal SYSDATE-1, meaning you’ve got all of yesterday’s records, but this process is horribly unreliable. Time-based data selection loads duplicate rows when it is restarted from mid-process failures. This means that manual intervention and data cleanup is required if the process fails for any reason.

Meanwhile, if the nightly load process fails to run and misses a day, a risk exists that the missed data will never make it into the data warehouse.

- **Full database "diff compare":** A full diff compare keeps a full snapshot of yesterday's database, and compares it, record by record against today's database to find what changed. It's good that you are guaranteed to find every change in this technique, but this technique is very resource intensive.

3.4.4 Data Transformation

Data transformation is converting data from a source data format into destination data format. The data transformation process typically consists of multiple steps and each step may perform schema and instance-related transformations (mappings). Importantly, transformation codes are generated to reduce the amount of self-programming. Thus, it is necessary to specify the required transformations in an appropriate language, e.g., supported by a graphical user interface. Various ETL tools offer this functionality by supporting proprietary rule languages. A more general and flexible approach is the use of the standard query language SQL to perform the data transformations. Moreover you can utilize the possibility of application specific language extensions, in particular user-defined functions (UDFs) supported in SQL: 99 [27].

The functionality of data transformation includes [9]:

- Removing unwanted data
- Converting to common data names and definitions
- Calculating summaries and derived data
- Establishing defaults for missing data
- Accommodating to source data definition changes.

3.4.5 Data Loading

After data has been extracted, it is time to be load into a data warehouse. The data which is cleansed and transformed to comply with the data warehouse standards is moved into the appropriate data warehouse entities. In this step, data may be summarized and reformatted as part of this process. This depends on the extraction and cleansing specifications and the performance requirements of the data warehouse. After the data has been loaded, metadata information is updated to reflect the activity that has just been completed [9].

3.4.6 Data Staging

To simplify the cleansing and transformation process, it may be practice creating and defining a staging area. This is a simple concept and allows maximizing up-time of a data warehouse while extracting and cleansing the data. A staging area is simply a temporary work area that data from source systems is copied and also it can be used to manage transactions that will be further processed to develop data. A staging area is mainly required for timing reasons. Briefly, in advance of data integration into the Data Warehouse, all required data must be available [9] [28].

3.4.7 Operational Data Store

For the need for integrated tactical and operational reporting database, the data warehouse was included a new database called the **Operational Data Store (ODS)** [29].

An ODS is an environment where data from different operational databases is integrated. It provides an integrated view of enterprise data to the users and enables the user to address operational challenges that span over more than one business function [30].

An ODS is subject to change much more frequently than a DW and stores, in contrast to a DW, no histories over operational data. Thus, an ODS provides support for activities such as collective operational decisions based on current company-wide information.

3.4.8 Data Integration

A data warehouse is an integrated collection of subject-oriented data in the support of decision making. Therefore, data integration is a core requirement of any data warehouse. The integration of data sources is mainly accomplished by the use of ETL processes. Hence, the appropriate designs of the ETL processes are key factors in the success of data warehouse projects [31].

3.4.9 Comprehensive Database

A comprehensive database platform is needed for data warehousing and business intelligence that combines scalability and performance, reliability, security, deeply integrated analytics, and embedded integration and data-quality.

Also it may provide an integrated platform for analytics; by embedding OLAP, Data Mining, and statistical capabilities directly into the database [32].

3.4.10 Metadata

Metadata is a kind of data that describes the data warehouse itself, in short data about data and it is a vital area of data warehouse. Metadata within a data warehouse describes and locates data components, their origins (which may be either the operational systems or the data warehouse), and their movement through the data warehouse process. The data access, data stores, and processing information will have associated descriptions about the data and processing documented in the metadata. This metadata should be managed from the beginning of data warehouse project. Information in the metadata repository includes [9]:

- The data model description
- Description of the layouts used in the database design
- Definition of the primary system managing the data items
- A data map, from the system of record to other locations in the data warehouse, including the descriptions of transformation and aggregations
- Database design definitions
- Data element definitions, including rules for derivations and summaries.

3.4.11 Middleware Tools (enable access to the DW)

The main purpose of data warehousing is to provide information to business users for strategic decision making and interaction with the data warehouse made by using front-end tools. Although, main delivery tools for analysis in a data warehouse are ad hoc requests, regular reports, and custom applications, many development efforts of data warehousing projects are focusing on exceptional reporting also know as alerts. The alerts warn a user when a certain event has occurred. For example, if a data warehouse is designed to assess the risk of stock decrease, an alert can be activated when a certain stock rate drops blow a predefined threshold. When an alert is well synchronized with the key objectives of business, it can provide warehouse users an enormous advantage. The front-end user tools can be grouped as follows [9]:

- Data query and reporting tools
- Application development tools

- Executive information systems (EIS) tools
- Online analytical processing (OLAP) tools
- Data mining tools.

3.5 Data Warehouse Development Approaches

Data Warehouse development approaches are evolves in time and today there are many approaches but two of them are very famous:

1. **Inmon Model:** Enterprise Data Warehouse (EDW) approach
2. **Kimball Model:** Data Mart approach

A comprehensive comparison summary of these models published by Breslin [33].

Bill Inmon suggests a **top-down** development approach. This approach adapts traditional relational database tools to develop needs of an enterprise wide data warehouse. Individual departmental databases are developed to serve most decision support needs from this enterprise data store [33].

Ralph Kimball, on the other hand, advocates a **bottom-up** development approach. This approach uses dimensional modeling, a data modeling approach unique to data warehousing. Kimball suggests creating one database (or data mart) per major business process, rather than building a single enterprise wide database. Enterprise wide cohesion is accomplished by using another Kimball innovation, a data bus standard [33].

Here, one may be ask which model is the best. Indeed, there is no one-size-fits-all strategy to data warehousing.

3.5.1 Inmon Model

Inmon's model is a top-down approach. Inmon's model consists of all information systems and their databases throughout a given organization and this is called the **Corporate Information Factory (CIF)** [34]. Figure 5 shows Inmon's Corporate Information Factory.

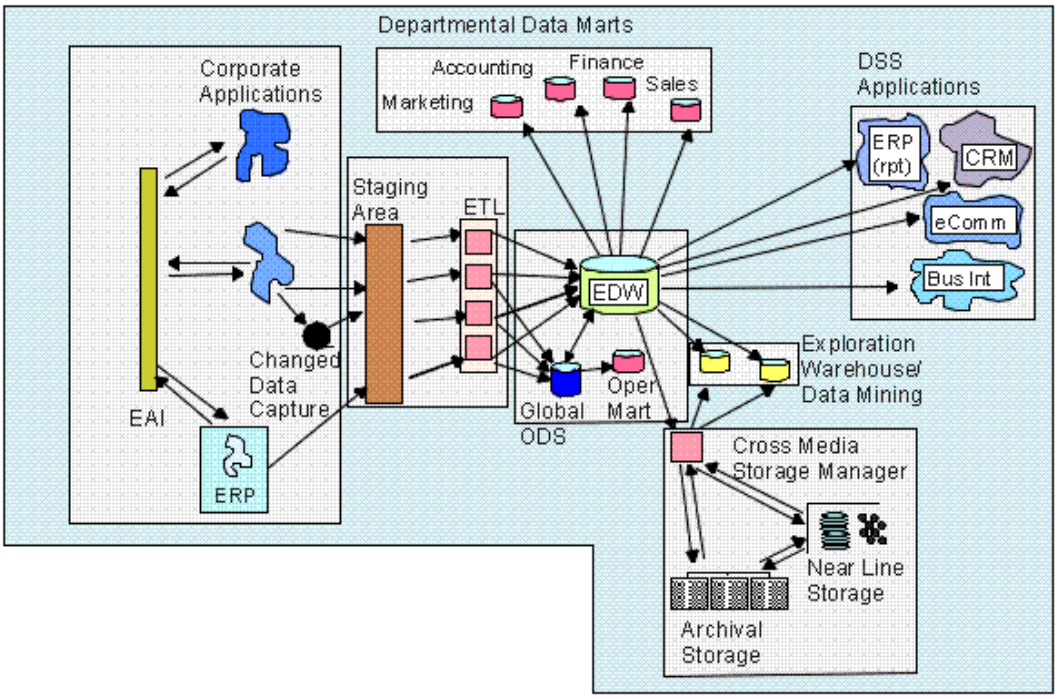

Figure 5: Corporate Information Factory [34]

In Inmon's CIF architecture; first there were applications, then online applications, followed by extract processing. After that a spider's web mess of data and systems came, then data warehouse and the Corporate Information Factory with its many architectural components [34]:

- The Operational Data Source
- The data marts environment
- Decision Support Systems
- Exploration/data mining warehouses
- Alternate storage, et al.

The overall database environment of the organization is divided into four levels by Inmon. [33]:

1. Operational
2. Atomic data warehouse
3. Departmental
4. Individual

The first level contains data from other transaction processing and legacy systems. The day-to-day operation of the organization is supported in this level; in

other words, the first level supports all transaction processing. Data from the operational systems is extensively manipulated and then moved to the atomic data warehouse. The last three levels comprise the data warehouse [35].

Inmon exemplifies the difference between operational data and data stored in the atomic data warehouse. In the example, the entity is a customer, and the attribute of most interest is the customer's credit rating. The operational system's database contains the customer's current credit rating and related information of interest such as loan balances, address, in a single record. By contrast, the atomic data warehouse contains the credit history for this customer, summarized by year, with one record per year [35].

The data contained in the departmental level is small to a great extend summarized. Each department's database can hold data summarized according to its information needs. In addition, Inmon's architecture ensures data consistency because all departmental data comes from the atomic data warehouse [35].

The fourth and final level of the architected environment is created by individual users. They create heuristic, ad hoc data sets as part of decision support analyses. This fourth level is housed on the individual user's personal computer and temporary [35].

It is possible to query the atomic data warehouse, if the department's database has not retained the data at the level of detail needed. Inmon says that the atomic data warehouse is worth the initial effort to construct because it allows the creation of any number of departmental databases without risking creating incompatible data between them [35]. A three-level data model is used for this.

3.5.1.1 The Three-Level Data Model

Inmon proposes three levels of data modeling.

1. ERD (Entity Relationship Diagram)
2. DIS (Data Item Set)
3. Physical

The first is just as in the development of operational databases. ERDs are used to explore and refine entities, their attributes, and the relationships between

entities. A set of ERDs are created for each department. The sum of all department ERDs is the corporate ERD [35].

The second data model consists of four constructs:

- A primary data grouping
- A secondary data grouping
- A connector, identifying the relationships of data between major subject area
- “Type of ” data

The second data model creates the DIS for each department. The corporate DIS is composed of the sum of the departmental DISs [35]. Figure 6 shows the relationship between Levels One and Two of Inmon’s Data Model.

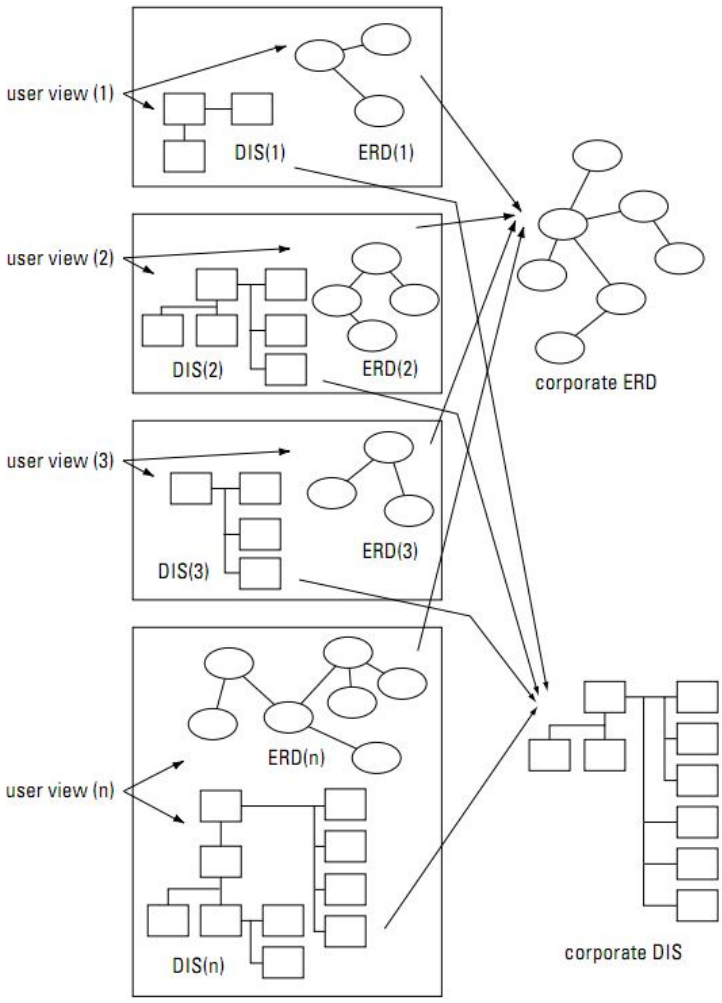

Figure 6: Relationship between Levels One and Two of Inmon’s Data Model [35]

The third and final level of Inmon's data model is the physical. "*The physical model is created from the mid-level data model merely by extending the mid-level data model to include keys and physical characteristics of the model*" [35]. Inmon explains various techniques (such as creating arrays of data, pre-formatting, rejoining tables etc.) for optimizing the performance of the data warehouse. The purpose is optimizing I/O performance that is the same as for operational database systems. Most of these techniques involve de-normalization of tables [33].

After the three-level data model is complete, the data warehouse development has begun by using Inmon's special adaptation of the spiral development methodology, which he calls Meth2. (Meth1 is for developing operational systems; Meth3 is for tuning an existing data warehouse). Inmon outlines ten steps, shown in Figure 7 [35].

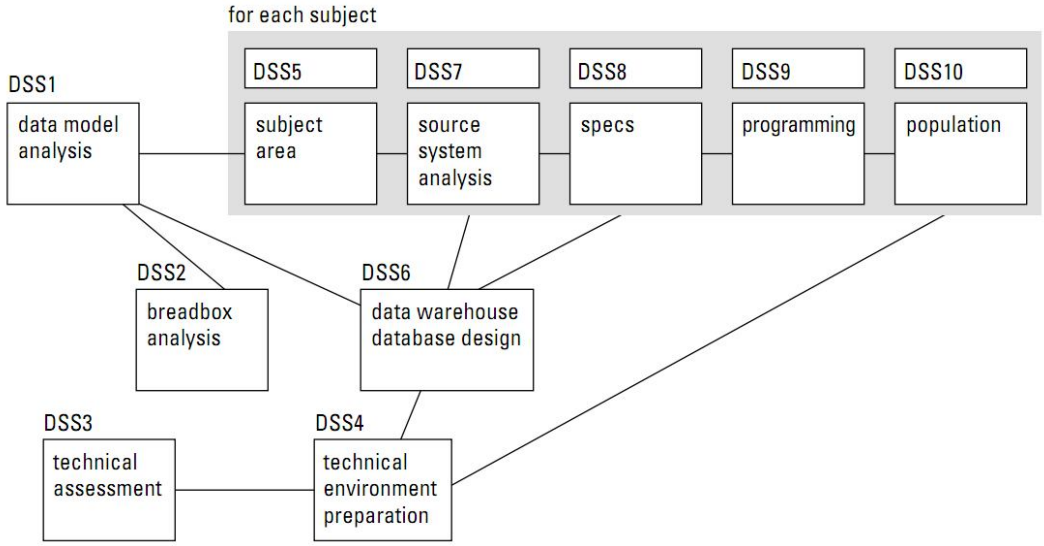

Figure 7: Inmon's Meth2 [35]

Inmon sees the data warehouse as an integral part of the Corporate Information Factory (CIF). This means that the data warehouse and operational databases are all part of a larger whole. Therefore, Inmon's data warehouse must adhere to most of the same standards as operational systems [35] [33].

3.5.2 Kimball Model

Kimball's model is a bottom-up approach. Kimball's model differs from a traditional relational database approach. One significant difference is that data

warehouses built with the Kimball model use a data modeling method which is called "***Dimensional Data Modeling***" [33].

The overall architecture features multiple databases that are expected to be highly interoperable. The data bus is the main design feature that makes this possible. This is another difference, "***The Data Bus and Conformed Dimensions***" [33]. The Data Bus and Conformed Dimensions is explained in section 3.5.2.1.

Dimensional modeling begins with tables. The tables are called either fact or dimension. Fact tables contain metrics, while dimension tables contain attributes of the metrics in the fact tables. Dimension tables routinely contain repeating groups; this violates normalization rules in order to achieve a high level of performance in the data warehouse [33]. Dimensional modeling is explained in section 3.6 en detail.

3.5.2.1 The Data Bus and Conformed Dimensions

In Kimball's model, data is copied from operational source systems to a staging area. The data is made consistent and suitable for end-user queries. From the staging area, data is loaded into data marts. The source of data for user queries are data marts [13].

Each data mart is based on a single business process such as point of sale, inventory, procurement, and order management. More than one department may be interested in a given business process. There is no one department that is perceived as the sole owner of a given data mart [13].

The bus architecture allows the sum of the data marts to be an integrated whole. That is, all data marts must use standardized conformed dimensions. Keys, column names, attribute definitions, and attribute values are consistent across business processes. This is the basic requirements of conformed dimensions. In other words, two dimensions are conformed "*when they are exactly the same, or one is a perfect subset of the other. Most important, the row headers produced in answer sets from two different conformed dimensions must be able to be matched perfectly*" [13]. This may seem an impossible set of requirements, but a knowledge of dimensional data modeling and adherence to the four-step dimensional design process help keep the requirements manageable [33].

A product dimension that spans multiple business processes may be an example. An artificial key assigned to the primary key for the product during the ETL process. The first data mart development defines the product key, and all subsequently developed data marts must use the key. Therefore, queries can be made across data marts without conflicting results [33].

3.5.2.2 The Four-Step Dimesional Design Process

Kimball suggests a development methodology that involves a bottom-up approach, which in the case of data warehouses means to build one data mart at a time. The four steps of the dimensional design process are:

1. Select the business process
2. Declare the grain
3. Choose the dimensions
4. Identify the facts

The first step is "select the business process" that has "*the most impact—it should answer the most pressing business questions and be readily accessible for data extraction*" [13].

The second step is declaring the grain that is the process of deciding what level of detail the data warehouse will contain. The lowest level of granularity is called atomic, that is, it cannot be further subdivided. Choosing a grain at the atomic level is highly important. If you choose a more summarized level, queries below that level cannot be fulfilled by the data warehouse [13].

The third step is to choose dimensions. Each of the dimension tables has a large number of attributes. The date dimension table includes many attributes such as Day, Week, Month so on [13].

The final step is to identify facts which to include in the fact tables. Kimball chooses to include some computed values in the fact table as well as truly atomic values. Therefore, this makes queries easy for the end user and provides acceptable data warehouse performance [13]. The result of the four-step process is shown in Figure 8.

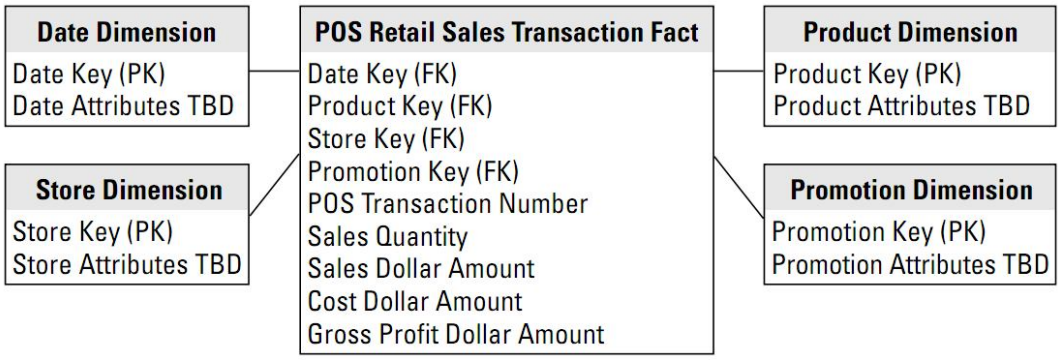

Figure 8: Sample Kimball Fact and Dimension Tables [13]

3.6 Dimensional Modeling

3.6.1 Entities within a Data Warehouse

A dimensional model such as star schema contains three types of logical entities: (1) a measure, (2) dimension, and (3) category detail. It is a logical structure which has measure entity at the center containing factual data, and this is surrounded by dimension entities containing reference data [9].

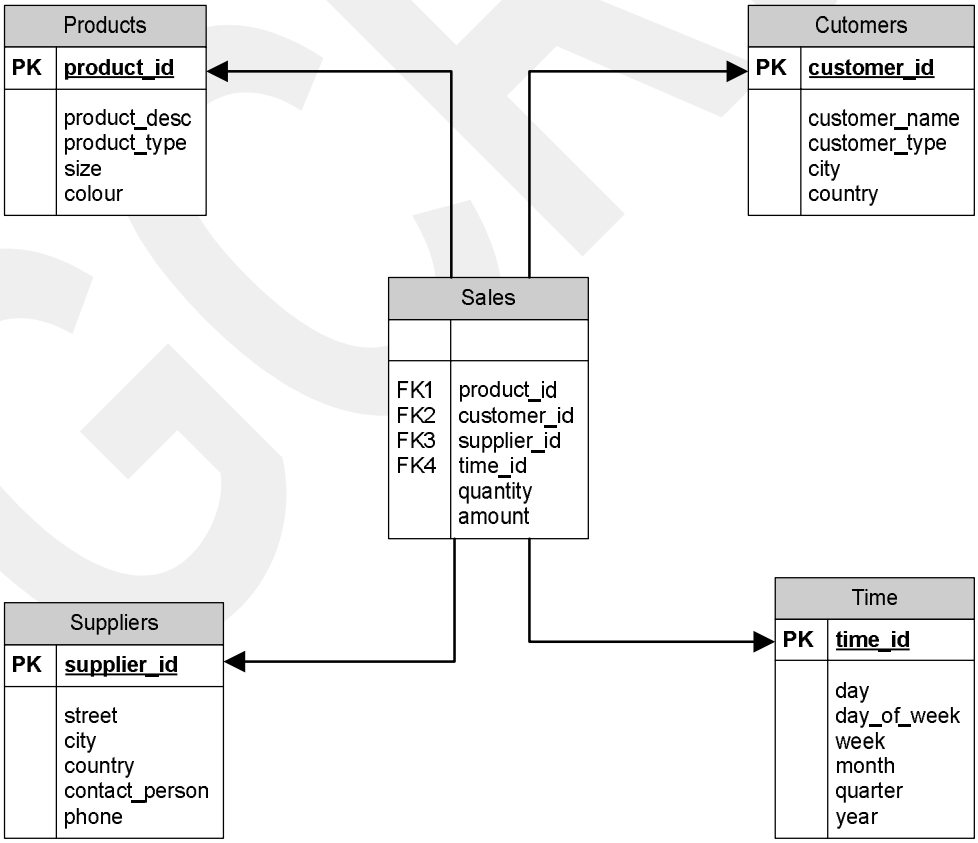

Figure 9: Sample Dimensional Model (Star Schema)

A dimensional model is shown in Figure 9. Sales table in the figure is a fact table; quantity and amount are measures of this fact table.

3.6.1.1 Measure Entities

Within a dimensional model (i.e. star schema), the center of the model – and often the focus of the users' query activity – is measure entity (or fact table). The data contained in a measure entity is factual information from which users derive business intelligence. The measurement data provides users with quantitative data about business [9].

3.6.1.2 Dimension Entities

Dimension entities are much smaller entities than measure entities. The dimension and their associated data allow users of data warehouse to browse measurement data with ease of use and familiarity. These entities assist users in minimizing the rows of data within a measure entity, and aggregating key measurements data. In this sense, these entities filter data, or force the server to aggregate data, so that fewer rows are returned from the measure entities [9].

3.6.1.3 Category Detail Entities

Each element in a dimension is a category, and represents an isolated level within a dimension that might be require more detailed information to fulfill a user's requirement. These categories that require more detailed data are managed within category detail entities. These entities have textual information that supports the measurement data and provides more detailed or qualitative information to assist in decision making process [9].

3.6.2 Star Schema

The **star schema** is the simplest data warehouse schema. It is called a star schema because the entity-relationship diagram of this schema resembles a star, with points radiating from a central table. The center of the star consists of a large fact table and the points of the star are the dimension tables [36].

A **star query** is a join between a fact table and a number of dimension tables. Each dimension table is joined to the fact table using a primary key to foreign key join, but the dimension tables are not joined to each other. The optimizer recognizes star queries and generates efficient execution plans for them. It is not

mandatory to have any foreign keys on the fact table for star transformation to take effect [36].

A typical fact table contains keys and measures. For example, in the Figure 9 sample schema, the fact table, sales, contain the measures quantity and amount, and the keys product_id, customer_id, supplier_id and time_id. The dimension tables are customers, times, products and suppliers. The products dimension table, for example, contains information about each product number that appears in the fact table.

A **star join** is a primary key to foreign key join of the dimension tables to a fact table.

The main advantages of star schemas are [36]:

- Provide a direct and intuitive mapping between the business entities being analyzed by end users and the schema design.
- Provide highly optimized performance for typical star queries.
- Are widely supported by a large number of business intelligence tools, which may anticipate or even require that the data warehouse schema contain dimension tables.

Star schemas are used for both simple data marts and very large data warehouses. Figure 10 presents a graphical representation of a star schema.

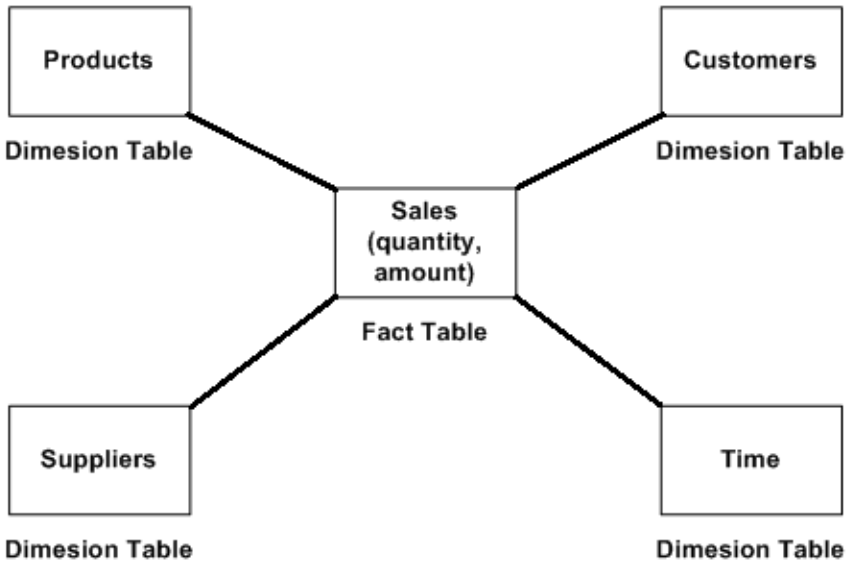

Figure 10: Star Schema

3.6.3 Snowflake Schema

The snowflake schema is a more complex data warehouse model than a star schema, and is a type of star schema. It is called a snowflake schema because the diagram of the schema resembles a snowflake [36].

Snowflake schemas normalize dimensions to eliminate redundancy. That is, the dimension data has been grouped into multiple tables instead of one large table. For example, a product dimension table in a star schema might be normalized into a products table, a product_type table, and a product_manufacturer table in a snowflake schema. While this saves space, it increases the number of dimension tables and requires more foreign key joins. The result is more complex queries and reduced query performance. Figure 11 presents a graphical representation of a snowflake schema.

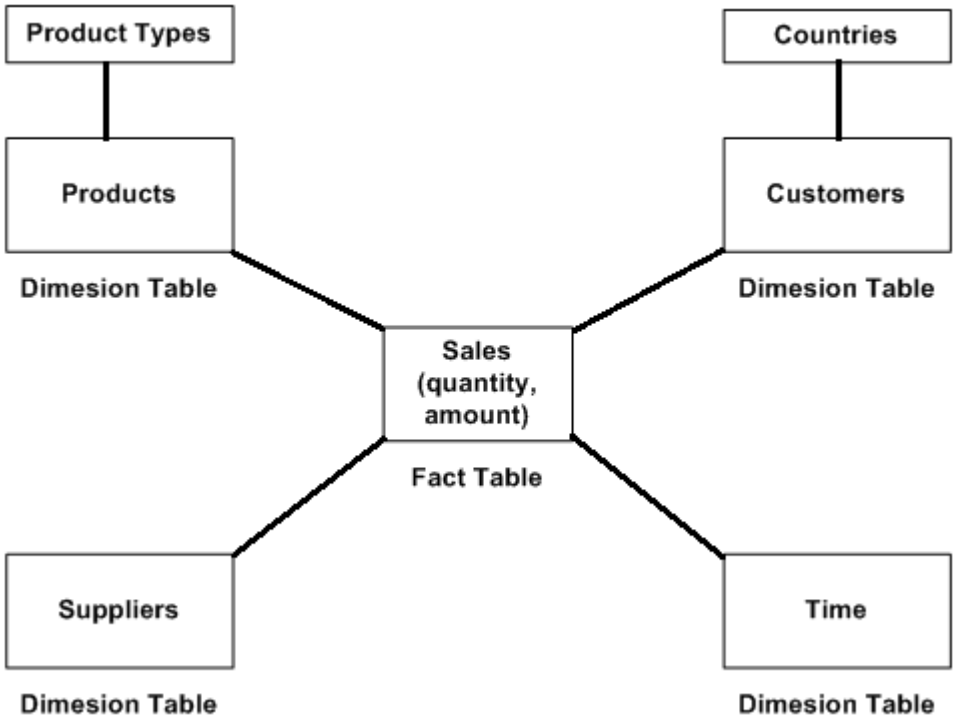

Figure 11: Snowflake Schema

3.6.4 Slowly Changing Dimensions

Entities such as customer demographics, product characteristics, classification rules, status of customers change over time. In a transaction system, many times the change is overwritten and track of change is lost.

For example a source system may have only the latest customer PIN Code, as it is needed to send the marketing and billing statements. However, a data warehouse needs to maintain all the previous PIN Codes as well, because we need to track on how many customers move to new locations over what frequency.

A key benefit for Data Warehouse is to provide historical information, which is typically over-written (and thus lost) in the transaction systems. How to handle slowly changing dimensions in a Dimensional Model is a key determinant to that benefit [37].

Dimension is a term that refers to logical groupings of data such as geographical location, customer information, or product information. **Slowly Changing Dimensions** (SCDs) are dimensions that have data that changes slowly, rather than changing on a time-based, regular schedule [13].

The SCD technique is used to preserve history in the Data Warehouse environment. The most common slowly changing dimensions are Types 1, 2, and 3.

3.6.4.1 Type 1

SCD Type 1 methodology overwrites old data with new data, and therefore does not track historical data at all. It is a simple overwrite of the existing dimension record. This means that no history will be preserved. This is most appropriate when correcting certain types of data errors, such as the spelling of a name [38].

For example, a database table that keeps supplier information. In this example, Supplier_Code is the "natural key" and Supplier_Key is a "surrogate key". Technically, the surrogate key is not necessary, since the table will be unique by the natural key (Supplier_Code). However, the joins will perform better on an integer than on a character string.

Now imagine that this supplier moves their headquarters to İstanbul. The updated table would simply overwrite this record:

Before Update:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_City
345	ASC	ABC Supply Co	Ankara

After Update:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_City
345	ASC	ABC Supply Co	İstanbul

The obvious disadvantage to this method of managing SCDs is that **there is no historical record kept in the data warehouse**. You can't tell if your suppliers are tending to move to the Midwest, for example. But an advantage to Type 1 SCDs is that they are very easy to maintain.

If you have calculated an aggregate table summarizing facts by city, it will need to be recalculated when the Supplier_City is changed.

3.6.4.2 Type 2

SCD Type 2 method tracks historical data by creating multiple records for a given natural key in the dimensional tables with separate surrogate keys and/or different version numbers. With Type 2, we have unlimited history preservation as a new record is inserted each time a change is made.

In the same example, if the supplier moves to İstanbul, the table could look like this, with incremented version numbers to indicate the sequence of changes:

Before Update:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_City	Version
345	ASC	ABC Supply Co	Ankara	0

After Update:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_City	Version
345	ASC	ABC Supply Co	Ankara	0
346	ASC	ABC Supply Co	İstanbul	1

Another popular method for tuple versioning is to add “effective date” columns.

After Update:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_City	Start_Date	End_Date
345	ASC	ABC Supply Co	Ankara	01.01.2008	21.12.2011
346	ASC	ABC Supply Co	İstanbul	22.12.2011	null

The null End_Date in row two indicates the current tuple version. In some cases, a standardized surrogate high date (e.g. 31.12.9999) may be used as an end date, so that the field can be included in an index, and so that null-value substitution is not required when querying.

Transactions that reference a particular surrogate key (Supplier_Key) are then permanently bound to the time slices defined by that row of the slowly changing dimension table. An aggregate table summarizing facts by state continues to reflect the historical state, i.e. the state the supplier was in at the time of the transaction; no update is needed.

If there are retrospective changes made to the contents of the dimension, or if new attributes are added to the dimension (for example a Sales_Rep column) which have different effective dates from those already defined, then this can result in the existing transactions needing to be updated to reflect the new situation. This can be an expensive database operation, so Type 2 SCDs are not a good choice if the dimensional model is subject to change.

3.6.4.3 Type 3

SCD Type 3 method tracks changes using separate columns. Whereas Type 2 had unlimited history preservation, Type 3 has limited history preservation, as it's limited to the number of columns designated for storing historical data. Where the original table structure in Type 1 and Type 2 was very similar, Type 3 adds additional columns to the tables. In the following example, an additional column has been added to the table so as to record the supplier's original city: (only the previous history is stored)

After Update:

Supplier_Key	Supplier_Code	Supplier_Name	Original_Supplier_City	Effective_Date	Current_Supplier_City
345	ASC	ABC Supply Co	Ankara	22.12.2011	İstanbul

Note that this record--having only a column for the original city and a column for the current city--can not track all historical changes, such as when a supplier moves a second time.

One variation of this type is to create the field Previous_Supplier_State instead of Original_Supplier_State which would then track only the most recent historical change.

3.6.5 OLAP Cube

Cubes are the logical storage structures that define a set of related dimensions. Each cell in the cube holds one value which is an intersection of the dimensions.

A cube view shown in Figure 12. The cube has three dimensions which are time, region and product. From the cube in figure, one can get the value(s) of cell(s). For example, in the figure, we gets the value of the cell which is the intersection of dimension values "City 1", "Product 4" and "January".

On a cube, each dimension enables you to perform specific OLAP operations. The basic OLAP operations are Roll up, Drill down, Slice, Dice, Pivot [14] [1].

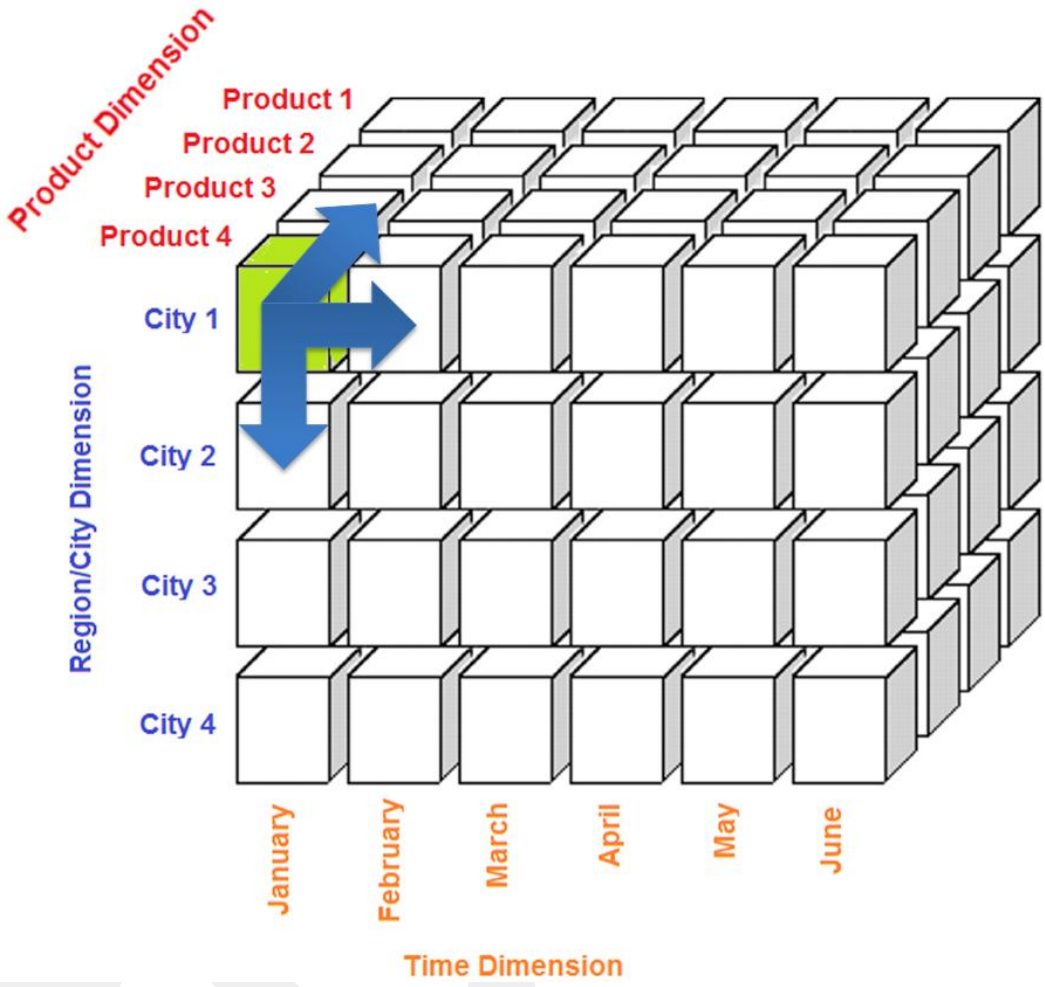

Figure 12: A 3D Cube View

CHAPTER 4

REAL TIME DATA WAREHOUSE

4.1 Introduction

Today's business environment is quickly changing and business decision makers need for a historical picture of what happened and a picture of what was happening today. Engineers strongly defended the notion that the data warehouse needed to provide a reliable information floor upon which to stand, providing an unwavering set of data to business decision makers. Because of the twinkling database, business users were directed to the production applications that run the business for up-to-the-moment reporting. Therefore, users had to go to the data warehouse for a historical picture of what happened in the business as of yesterday and had to look across many OLTP systems for a picture of what was happening today. This division never fully accepted by business users. They want to go to one place to get the business information that they needed [19].

Fresh data in data warehouses is a strong feature from the part of the users. Traditionally, loading data into warehouses has been performed in an off-line period. In such a data warehouse setting, data are extracted from the sources, transformed, cleaned, and loaded to the warehouse. To avoid overloading the source production systems, data warehouse activities takes place during a loading window, usually during the night. In most cases, a data warehouse is typically updated every day (24 hours period) [39].

The delay between a business transaction and its appearance in the data warehouse is too much for many organizations in fast-moving vertical industries. Additionally the data warehouse has become mission critical. That is, feeding enriched information back to operational systems that is then used to process

transactions; personalize offers, and present up-sell promotions. The push for ever-fresher information is needed [19].

The issues facing managers and government officials in today's business environment is rapidly making historical systems less valuable. Decisions in the business world become more real-time and the systems that support those decisions need to keep up to date. It is solely natural that Data Warehouse, Business Intelligence, Decision Support, and OLAP systems quickly begin to include real time data. "*Morning sales on the east coast will affect how stores are stocked on the west coast. Airlines and government agencies need to be able to analyze the most current information when trying to detect suspicious groups of passengers or potentially illegal activity. Fast-paced changes in the financial markets may make the personalized suggestions on a stockbroker's website obsolete by the time they are viewed*" [40] [41].

During the last five years business users are requesting more freshness. To give an example, a case study for mobile network traffic data, involving around 30 data flows, 10 sources, and around 2TB of data, with 3 billion rows [42]. In that case study, it is reported that user requests indicated a need for data with freshness at most 2 hours. However, business user requirements are getting more pressing as the time passes [39].

Today, there are new types of sources. The Web is considered as a source in many applications. In such a case, the notion of transaction at source side becomes more flexible and the data that appear at a source web site are not always available later; therefore, if reaction to a change is not taken instantly, important information, possibly, will not be gathered later, by the off-line refreshment of the warehouse. At the same time, business necessities - e.g., increasing competition, need for bigger sales, better monitoring of a customer or a goal, precise monitoring of the stock market, and so on - result in a demand for accurate reports and results based on current data and not on their status as of yesterday [39].

Another important issue that questions the conventional way of thinking about ETL is the globalization of the economy and the commodity trading business. The usual process of ETL-ing the data during the night in order to have updated reports in the morning is getting more complicated if we consider that an organization's branches may be spread in places with totally different time-zones.

Based on such facts, data warehouses are evolving to “active” or “live” data producers for their users, as they are starting to resemble, operate, and react as independent operational systems. In this setting, different and advanced functionality that was previously unavailable (for example, on-demand requests for information) can be accessible to the end users. For now on, the freshness is determined on a scale of minutes of delay and not of hours or a whole day. As a result, the traditional ETL processes are changing and the notion of “real time” or “near real time” is getting into the game. Less data are moving from the source towards the data warehouse, more frequently, and at a faster rate [39].

The traditional data warehouses are implemented using batch driven approach and mainly according to the pull technology principle. The data loading from source systems to data warehouses is generally performed on a nightly basis or even in some cases on a weekly basis; therefore typical data warehouses normally do not have the most current data [39]. Furthermore the operational systems may have to be go offline during the data extraction process. It is an unacceptable situation which generates delays in businesses especially that require instantaneous access to up-to-date information [44].

The way in which the data is brought into the data warehouse is extremely important and keeping the data in the warehouse closely synchronized with data from source databases is the most effective approach [45].

A real time data warehouse eliminates the data availability gap and enables organizations to concentrate on processing their valuable data. Furthermore, continuous data processing without delay opens up significant new opportunities [46].

The zero-latency enterprise is ideal for a business. This ideal urges the benefits of speed and a single version of the truth. In a real time, generally mean, information is delivered to the right place at the right time for maximum business value. We may call these *right-time* systems. At present, true zero latency is an unattainable ideal—it takes some time to synchronize information across several production systems and data warehouse—but there is a pressure on many modern data warehouses to provide a low-latency view of the business [19].

In summary, we may list three major reasons why we need a Real Time in a Data Warehouse:

- Faster reaction time
- Reduced decision time
- New process capabilities.

4.2 Real Time Data Warehouse Requirements

Real Time Data Warehouse architecture consists of a lot of views, approaches. Therefore, before building RTDW we have to analyze our requirements clearly from a real time needs perspective. Here, we will present some of these requirements [19]:

4.2.1 Data Freshness and Historical Needs

The developmental costs and complexity for reducing latency between operational databases (OLTPs) and the data warehouse obey the law of diminishing returns, lowering latency increases complexity and cost in a nonlinear fashion. Therefore, realistic goals and expectations about the freshness of the data warehouse are needed to set. So, for the needs of real time data warehouse it will be good to care the following considerations [19]:

- **Less than five minutes of latency.** This low latency cannot be reliably met through mainstream real time data warehousing. It always takes some nontrivial amount of processing and time to move, transform, and load information from the OLTP systems to the real time partition.

- **Single data source requirements demanding little or no history.** The reports that require none of the integrated and historical data features provided by the data warehouse are best addressed through the operational system itself. In fact, they should present a very small reporting overhead on the operational systems and should not degrade transactional performance significantly.

- **Reports with an entirely different audience from that of the existing data warehouse.** These types of reports might demand new reporting vocabularies and mechanisms for dissemination, and also factors that can overly complicate an already complex real time data warehousing development effort. Therefore the real time architect should be consider that business vocabularies and metrics.

- **No real need for ad-hoc analysis.** For ad-hoc analysis of the low-latency part of data, if there is little need, there may be no need a full-blown streaming ETL system redesign.

4.2.2 Reporting Only or Integration Also

This is about the need of the organization that if need of the organization are a one-way solution for moving operational data into the data warehouse for reporting purposes only, or are there also requirements for closing the loop by moving conformed dimension data between operational applications themselves and/or the data warehouse.

In fact, as an example, any strategic CRM initiative is wants the timeliest and most complete customer information available, which includes both operational customer data (information about recent sales or complaints, for example) and data-warehouse or data mining-derived customer marketing information such as customer segmentation, profiling, and lifetime value. This can be called True CRM [19].

4.2.3 Just the Facts or Dimension Changes Also

Facts and dimensions are important by the view of business people and dimensional data warehouse architects, but OLTP systems do not make such sharp distinctions. So that must be understood and categorized the OLTP business transactions of interest and designed appropriately. Are the real time report requirements focused exclusively on fresh facts or are they also concerned with fresh dimension transactions? If real time dimensional changes are needed for reporting, are they slowly or rapidly changing? [19].

4.2.4 Alerts, Continuous Polling, or Nonevents

An ETL system usually has a well-defined boundary where dimensionally prepared data is handed to the front room. A real time system cannot have this boundary in many cases. Also, the architecture of front-end tools is affected. There are three data-delivery paradigms that require an end-to-end perspective reaching all the way from the original source to the end user's screen [19]:

- **Alerts:** "*A data condition at the source forces an update to occur at the user's screen in real time*."

- **Continuous polling:** "*The end user's application continuously probes the source data in order to update the screen in real time.*"

- **Nonevent notification:** "*The end user is notified if a specific event does not occur within a certain time interval or as the result of a specific condition.*"

In each of these cases, the real time ETL system is communicate with the end user's application, either by sending a notification or by receiving a direct request.

4.2.5 Data Integration or Application Integration

You need to categorize your requirement as either data integration or application integration. In general, **data integration** that can be satisfied by simply moving data between databases. However, **application integration** (sometimes also called functional integration) can be described as assembling applications together through the use of some common middleware [19].

4.2.6 Point-to-Point versus Hub-and-Spoke

An important factor in selecting architecture is the number of publishing and subscribing systems that supports in the foreseeable future. If real time data warehouse is also supporting some degree of application (or functional) integration, this number can help to decide if a relatively simple point-to-point solution will suffice or if a more robust hub-and-spoke architecture will be required [19].

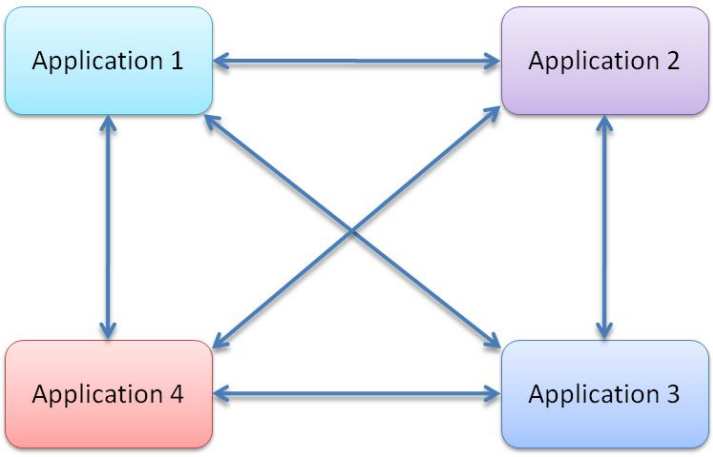

Figure 13: Point-to-Point Application Integration

Figure 13 shows a point-to-point application integration example. In this example, a small number of applications are exchanging data but point-to-point solutions can demand a very large number of data-exchange interfaces. Each of the interfaces requires maintenance whenever its source or target applications change.

In contrast to point-to-point architectures, a hub-and-spoke integration approach can be minimizing the number of customer interfaces and cross-system dependencies as seen on Figure 14.

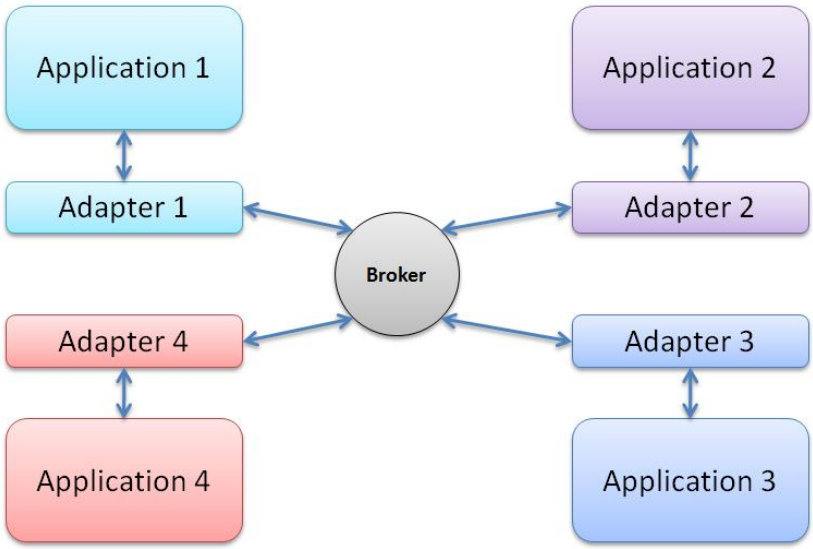

Figure 14: Hub and Spoke Application Integration

4.2.7 Data Cleanup Considerations

If the organization needs real time cleanup and synchronization of data, some additional factors in selecting an approach need to be considered. This often falls upon the data warehouse customer dimension manager to provide that ensures that no redundant data are created for the enterprise. It may be appropriate for the real time dimension manager to assume responsibility for matching (de-duplicating) records [19].

4.3 How Real Time Data Requirements Change Data Warehouse Environment

Most real time need will be driven by operational decision making, not strategic decisions [47].

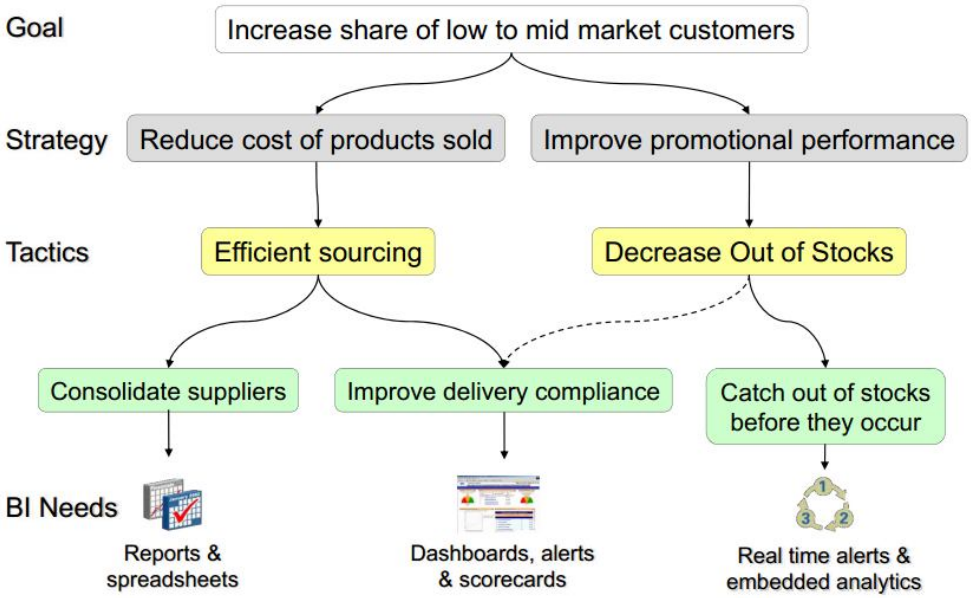

Figure 15: Strategy, Decisions and Data Latency [47]

Madsen (2008) briefly lists as follows (Table 2) which decisions benefit:

Table 2: Which Decisions Benifit

	Strategic	**Operational**
Decision Time	Flexible, Long Cycle	Constrained, Short Cycle
Decision Scope	Broad, Organizational	Narrow, Departmental or Process
Decision Model	Complex	Simple
Decision Latency	High, History is the Core to Decisions	Low, Recent Data is Core to Decisions
Data Scope	Many Sources, Many Types, Aggregated	Few Sources, Structured, Detailed

In a Data Warehouse we mostly handle big data. Definitions of big data focus on the size of data in storage but there are other important attributes of big data, namely data variety and data velocity. As seen in Figure 16, the three Vs of big data (volume, variety, and velocity) constitute a comprehensive definition and each of the three Vs has its own ramifications for analytics [48].

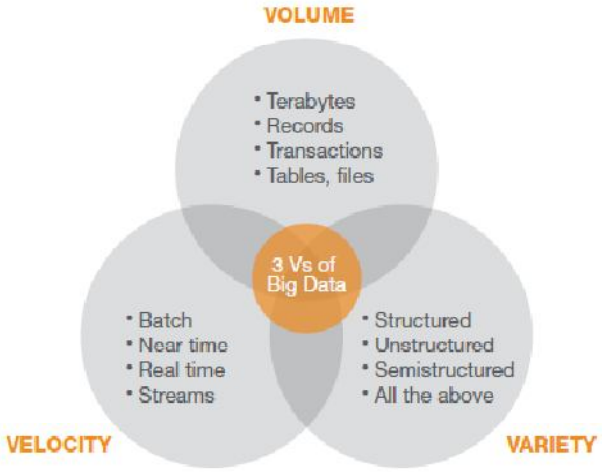

Figure 16: Three Vs of Big Data [47]

A number of unique challenges and opportunities come with real time data warehousing. These can be describing in two points of view: One is from a technical architecture perspective, and other is a data architecture perspective.

First one has the potential to change the big-bang approach. Big-bang approach needed during the nightly batch ETL load windows to a continuous ETL-like flow throughout the day. System-availability requirements may intensify the business comes to rely on low-latency availability of business transactions in the data warehouse.

The latter is that real time data warehousing challenges the posture of the data warehouse as system of discrete periodic measurements—a provider of business snapshots—advocating instead a system of more comprehensive and continuous temporal information. If the frequency of fact loading increases from once per day to every 15 minutes, but more dramatically if the loading of facts and dimension records occurs continuously. The data warehouse might then capture the business transactions and their dimensional context at all points in time. Slowly changing dimensions become rapidly changing dimensions. The data warehouse becomes more operational in nature [19].

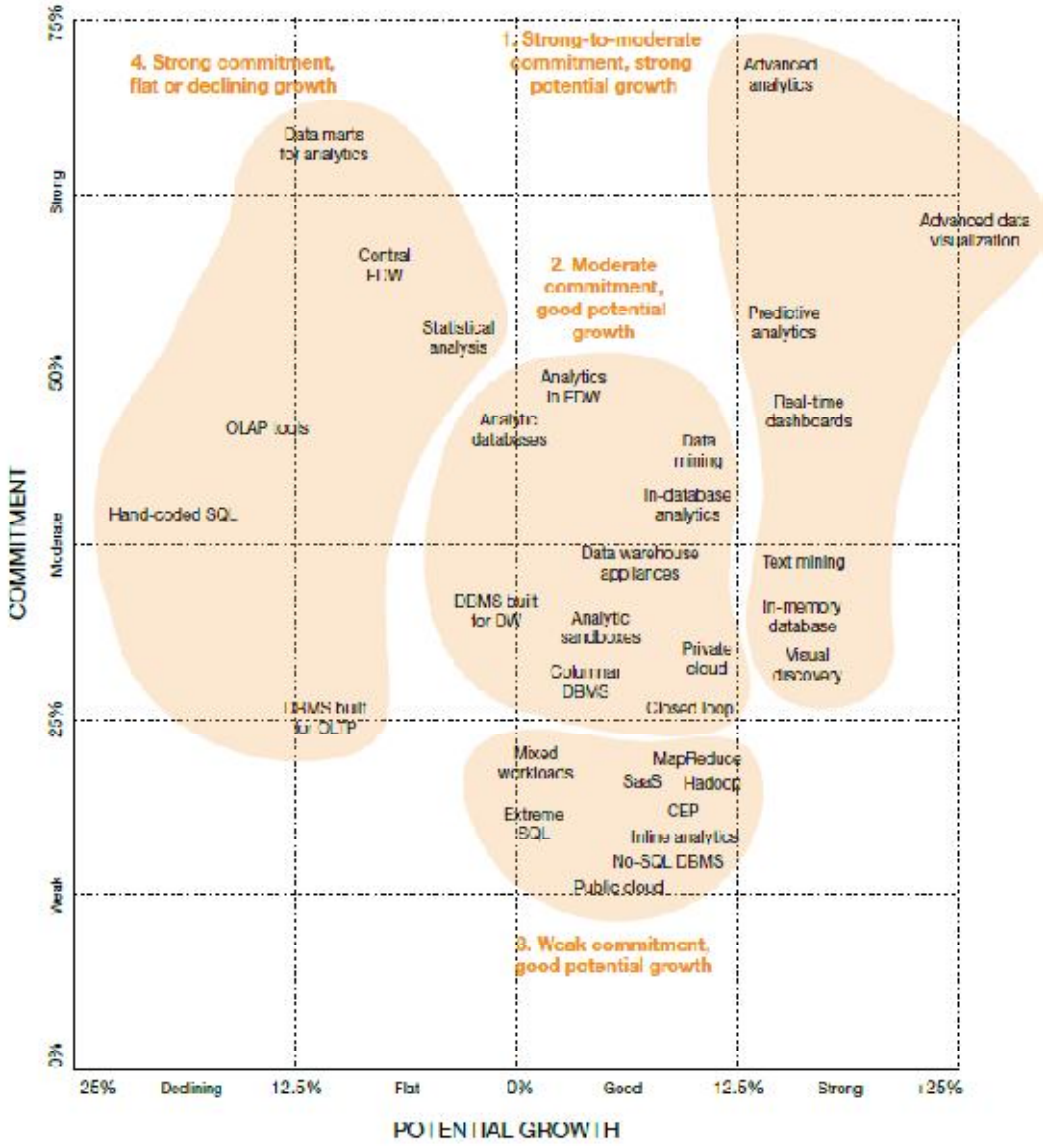

Figure 17: Options for Big Data Analytics Plotted by Potential Growth and Commitment [48]

4.4 Real Time ETL

Real time ETL is a miscall for a category of data warehousing services that is neither true real time nor, in many cases, but this term is mostly used in this area and so we used here. Instead, the term refers to that data moves asynchronously into a data warehouse with some urgency—within minutes of the execution of the business transaction. In many cases, real time data warehousing approaches are quite different from the ETL methods used in batch-oriented data warehousing. Basically, executing traditional ETL batches on an ever-more frequent schedule throughout the day might not be practical, either to the OLTP systems or to the data warehouse. Conversely, including the data warehouse in the OLTP system's

transaction commit logic cannot work either. The OLTP system does not have the luxury of waiting for the data warehouse loading transaction to commit before it proceeds with its next transaction, nor is any locking or two-phase commit logic practical across systems with different structures and different levels of granularity. Instead, you aim simply to move the new transactions into a special **real time partition** of the data warehouse within some timeframe acceptable to the business, providing analytic support for day-to-day operational decisions [19].

4.5 Real Time ETL Approaches

Addressing real time data warehousing requirements, a number of technologies are available. Here, we will discuss some of these technologies.

4.5.1 Microbatch ETL

A data warehouse can only be considered real-time, or near real-time, when all or part of the data is updated, loaded or refreshed on an intra-day basis, without interrupting user access to the system. Traditional ETL is useful for addressing daily, weekly, and monthly batch reporting requirements. Micro batch ETL designed for real-time data acquisition from an operational data source. Figure 18 and Figure 19 give a representation of the system. All of the new or changed transactions are captured as point-in-time snapshots for each load and moved to the data warehouse. Thus, changes to dimensions that occur between batch processes are lost in the warehouse. Therefore, we may say that ETL is not a suitable technique for data or application integration for organizations needing low latency reporting or for organizations that need more detailed dimensional change capture. But traditional ETL is a simple, direct, and tried-and-true method for organizations that have more casual latency requirements and complex integration challenges. Figure 18 shows the traditional ETL process [19] [48].

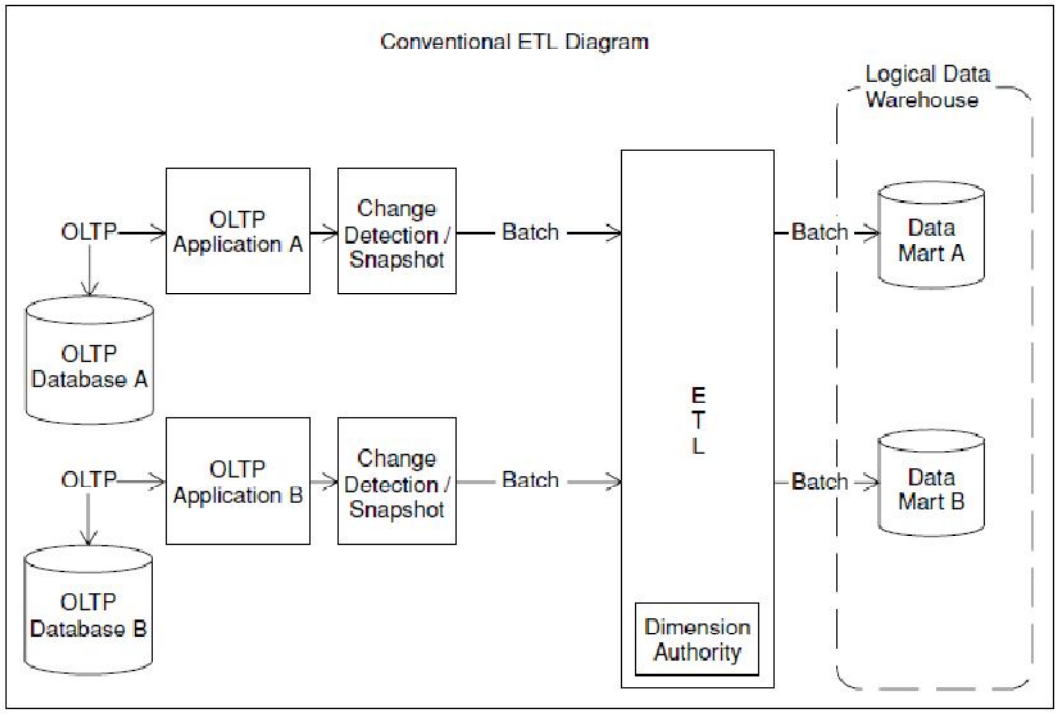

Figure 18: Traditional ETL Diagram [19]

Micro-batch ETL is similar to traditional ETL. The frequency of batches is increased in micro-batch ETL. These frequent micro batches are run through another ETL process and directly feed the real time partitions of the data marts. The real time partitions are moved to the static data marts and are emptied, once each day. Figure 19 shows a diagram of micro-batch ETL [19].

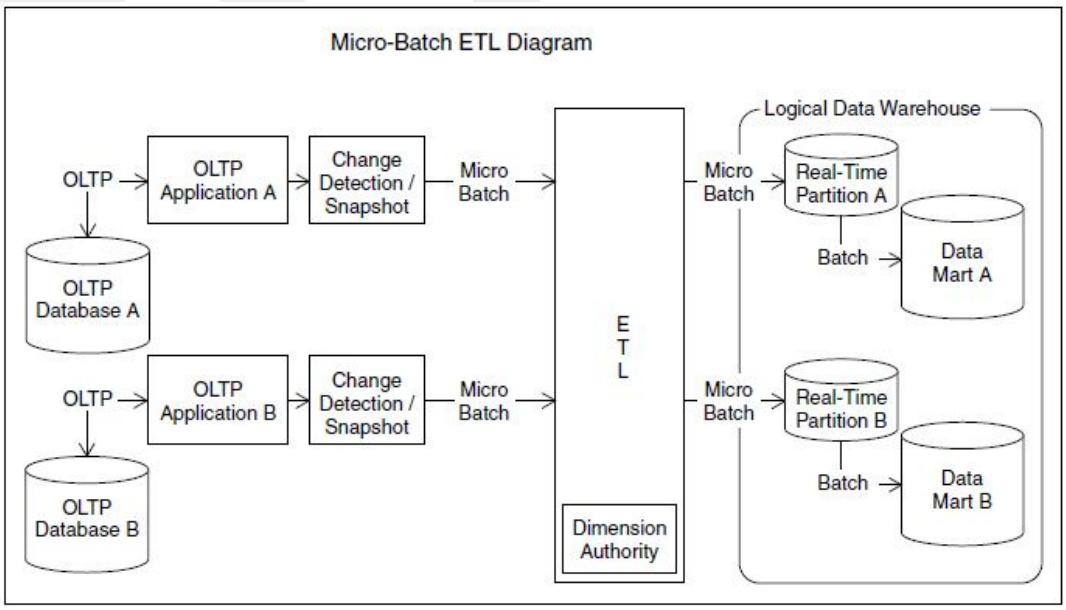

Figure 19: Micro-Batch ETL Diagram [19]

Slowly changing dimensions may become rapidly changing and grow deep due to the increased run frequency. Micro-batch ETL also demands a comprehensive job control, scheduling, dependency, error-mitigation method, metadata management and capable of executing data warehouse publication strategies in the face of most common data-loading issues. Additionally, micro-batch ETL demands more frequent detection of new and updated transactional records on the OLTP systems [19].

4.5.2 Enterprise Application Integration

Enterprise Application Integration (EAI) is an integration framework and it is composed of a collection of technologies and services that support true application integration, allowing individual operational systems to interoperate in new and potentially different ways than they were originally designed EAI form a middleware to enable integration of systems and applications across the enterprise, and sometimes called functional integration [19] [50].

EAI typically composed of a set of adapter and broker components that move business transactions. Communication across the various systems in the integration network is in the form of messages. Application specific adapters are responsible for dealing with all of the logic which includes create and execute messages. Brokers are responsible for routing the messages appropriately, based on publication and subscription rules [19].

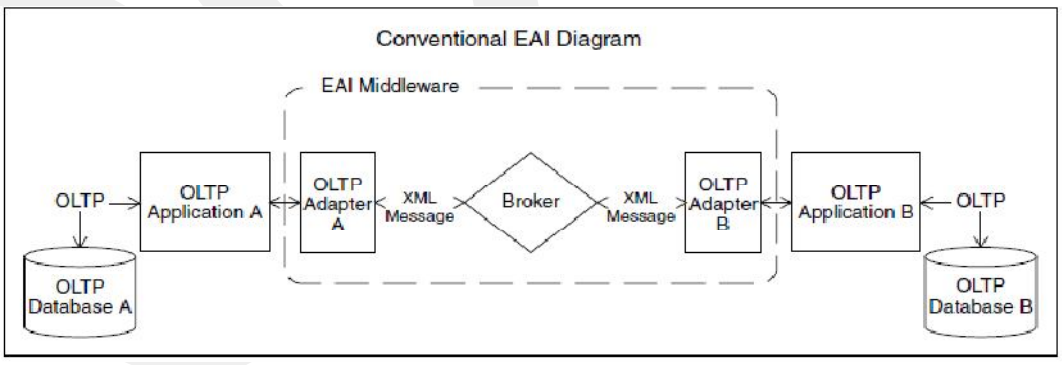

Figure 20: Convensional EAI Diagram [19]

EAI technologies can be strong enabling vehicles for the real time data warehouse. They support the ability to synchronize important data across applications and provide an effective means for distributing data-warehouse-derived information assets, such as new customer segmentation values, across the enterprise [19].

There are two fundamental components of EAI [49]:

- **Adapters:** The purpose of this component is to hide heterogeneity and present a uniform view. In other words, it map heterogeneous data formats, interfaces and protocols into a common model and format.
- **Message Broker:** This component facilitates interaction among adapters and, therefore, among the back-end systems that need to be integrated.

The real time EAI data warehouse architecture changes the monolithic ETL block. The dimension manager system(s) pulled out as separate architectural components, each with its own adapters, and placing responsibility for most of the transformation and loading chores of the data mart real time partitions on the data mart adapters. Any data-change transaction would be captured from the OLTP application by an adapter. After that adapter sent data as a non-conformed dimension message to the broker, which then routes it to whichever systems subscribe to non-conformed dimension messages. The dimensional record is conformed by the dimension manager. Then dimension manager adapter sends it back as a conformed dimension message to the broker, which then forwards it to all systems that subscribe to conformed dimension data, typically the OLTP systems and data marts [19]. Figure 21: Real Time DW/EAI Example Figure 21 is a diagram of a real time EAI data warehouse.

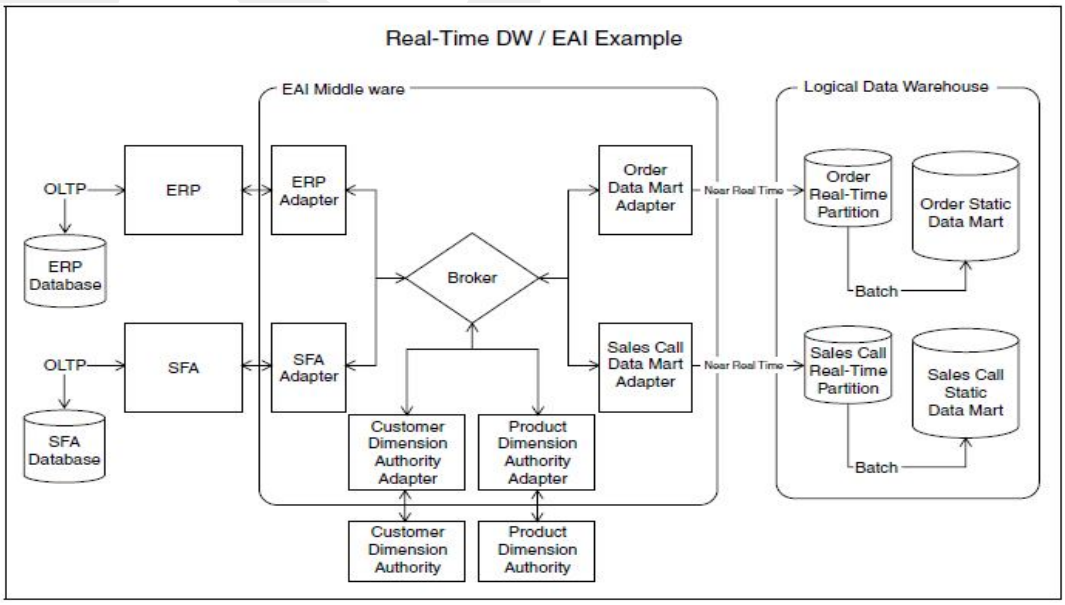

Figure 21: Real Time DW/EAI Example [19]

4.5.3 Capture, Transform, and Flow

Capture, Transform, and Flow (CTF) is a relatively new category of data integration tools. The aim is simplifying the movement of real time data across heterogeneous database technologies. The application layer of the transactional applications is bypassed and direct database-to-database exchanges are executed. Transactions, both new facts and dimension changes, can be transferred directly from the operational systems to the data warehouse staging tables with low latency, typically a few seconds. The transformation functionality of CTF tools is typically basic in comparison with today's mature ETL tools. Once data is staged, additional transformations beyond the capabilities of the CTF tool can be applied either by microbatch ETL or via triggers that fire on INSERT in the staging area. In either transformation scenario, records are then written directly into the real time partition tables. CTF can offer a compelling blend of the some of the benefits of EAI, while avoiding much its complexity [19]. Figure 22 diagrams CTF.

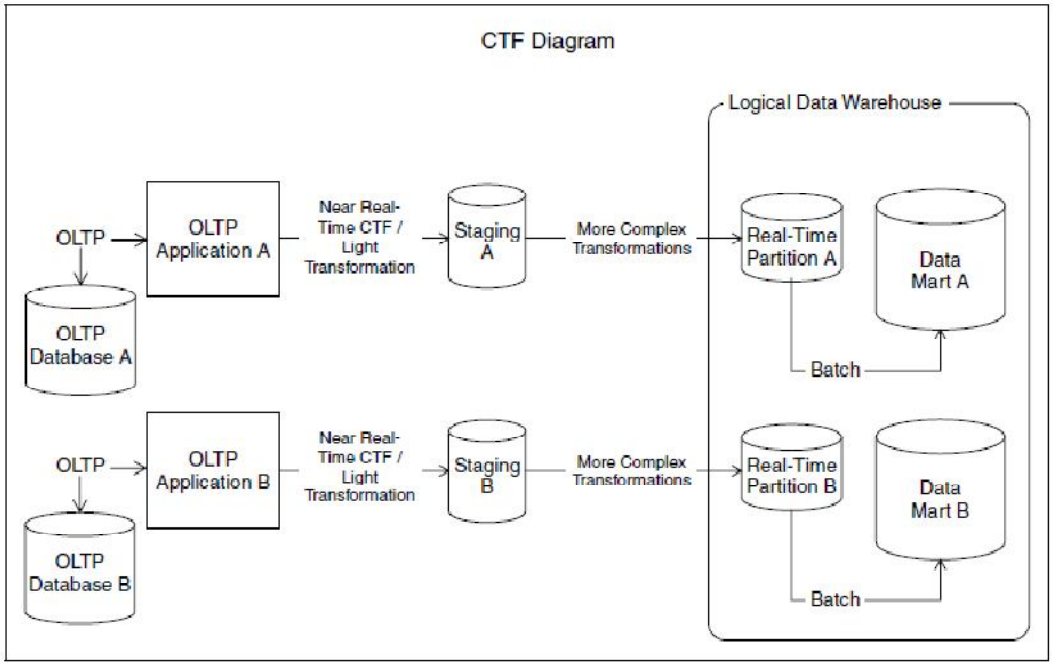

Figure 22: CTF Diagram [19]

4.5.4 Enterprise Information Integration

Enterprise Information Integration (EII) is another relatively new category of software. The aim is quickly adding real time reporting capabilities to business-intelligence systems. In this architecture, the logical view of the current data in the OLTP systems are presented to the business user in a structure appropriate for

analysis, and delivered on the fly via inline ETL transformation. It can be sensed a virtual real time data warehouse [19].

EII operates in somewhat similar conventional data warehouse mechanisms, except that instead of a data warehouse, the target might be a report, spreadsheet, or OLE DB or XML object. The EII system actually generates a series of queries, typically via SQL, at the moment requested, and then applies all specified transformations to the resultant data, and delivers the results to the business user. EII can be used as an effective data warehouse prototyping device and may be a compelling choice for organizations that need real time integrated operational reporting as quickly as possible [19].

4.5.5 The Real Time Dimension Manager

The real time dimension manager is used on converting incoming customer records into conformed customer records. This data may be incomplete, inaccurate, or redundant. Conformed means that dimensional records are turn into the best form that the organization is capable of achieving. A general diagram of the real time dimension manager is presented in Figure 23 [19].

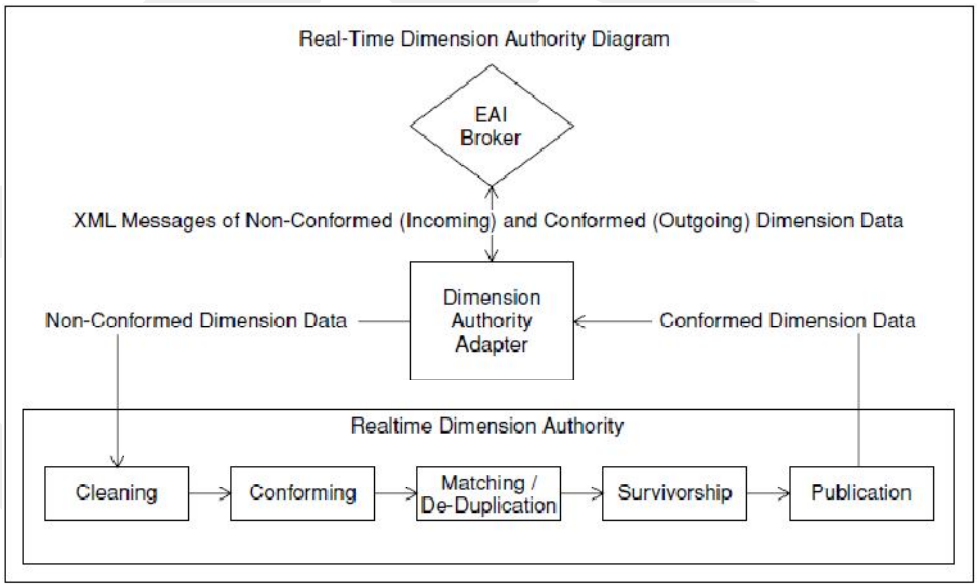

Figure 23: Real Time Dimesion Authority Diagram [19]

4.5.6 Microbatch Processing

There is a common dilemma when designing real time data mart partition or dimensional systems. Should the solution comprise straight through processing or utilize more frequent microbatches? One answer to the conflicting demands may

be frequent microbatches that give near real time performance within constraints imposed by batch-oriented toolsets. As opposed to using straight-through processing, they can be developed independently, have individually tunable batch-sizing specifications, and be replaced and/or upgraded independently as new toolsets with more features. Additionally, new processes such as specialized address verification or creditworthiness scoring can be more easily inserted into the job stream, and jobs that require selective processing are more easily accommodated. But this flexibility comes at a cost such as each process have defined and persistent interfaces, additional I/O [19].

4.6 Choosing an Approach

Selecting an appropriate architecture and approach is a very difficult task. Because, there are so many technologies to choose from, surrounded by so much vendor and analyst hype, and with so few successful case studies from which to draw best practices. The following table attempt to cut through some of this uncertainty by distilling some of the information into guidelines to narrow options. Table 3 is a comparison matrix of the presented approaches for real time reporting [19].

Table 3: Real Time Reporting Decision Guide Matrix [19]

		EII ONLY	EII + STATIC DW	ETL	CTF	CTF-MB-ETL	EAI
		ENTERPRISE INFORMATION INTEGRATION IN PACE OF REAL TIME DW	ENTERPRISE INFORMATION INTEGRATION IN CONCERT WITH CONVENTIONAL NON REAL TIME DW	STANDART ETL PROCESSING	CAPTURE TRANSFORM FLOW WITH ETL FEEDING REAL TIME DW	MICRO-BATCH ETL FEEDING REAL TIME DW	ENTERPRISE APPLICATION FEEDING INTEGRATION REAL TIME DW
Historical Data Supported		✓	✓	✓	✓	✓	
Reporting Data Integration Complexity	**Low**	✓	✓	✓	✓	✓	✓
	Moderate	✓	✓	✓	✓	✓	✓
	High	✓	✓	✓		✓	✓
Data Freshness / Maximum Latency	**1 Minute**	✓	✓				✓
	15 Minutes	✓	✓		✓		✓
	1 Hour	✓	✓		✓	✓	✓
	1 Day	✓	✓	✓	✓	✓	✓

CHAPTER 5

WEB SERVICES BASED REAL TIME DATA WAREHOUSE

5.1 Web Services and its Architecture

Web Services is standard set of open technical specifications and developed by the W3C. It has self-contained, self-describing and modular features. It can be used web publishing, search and call. Once Web Services configured, the other applications and Web Services can be discovered and invoked the service directly [51].

Web Services architecture composed of three roles, that is, service providers, service registry and service consumer. Web Services architecture based on the interaction between these three roles. Service provider defines the service description, and publishes it to the service registry. Service consumers (or service lookup operation from the local registry) search service description, then use the service description to bind with the service provider, and call the appropriate Web Services. Figure 24 shows these operations, these operations provide the components and their interaction [52].

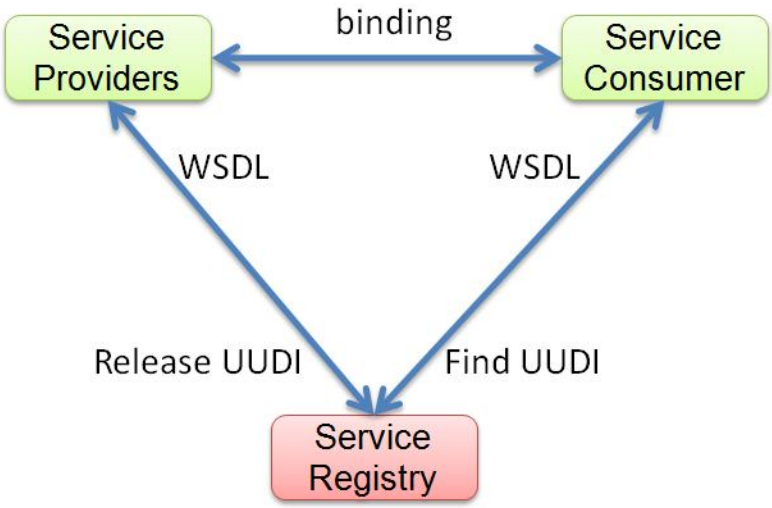

Figure 24: Web Services Architecture Model [52]

5.2 Web Services Based Real Time Data Warehouse Architecture

In this thesis study, we developed web services based real time data warehouse architecture, as shown in Figure 25. Components of this architecture as follows:

- Web Service Client
- Web Service Provider
- Metadata
- ETL
- Real Time Partition
- Data Warehouse
- Real Time Data Integration

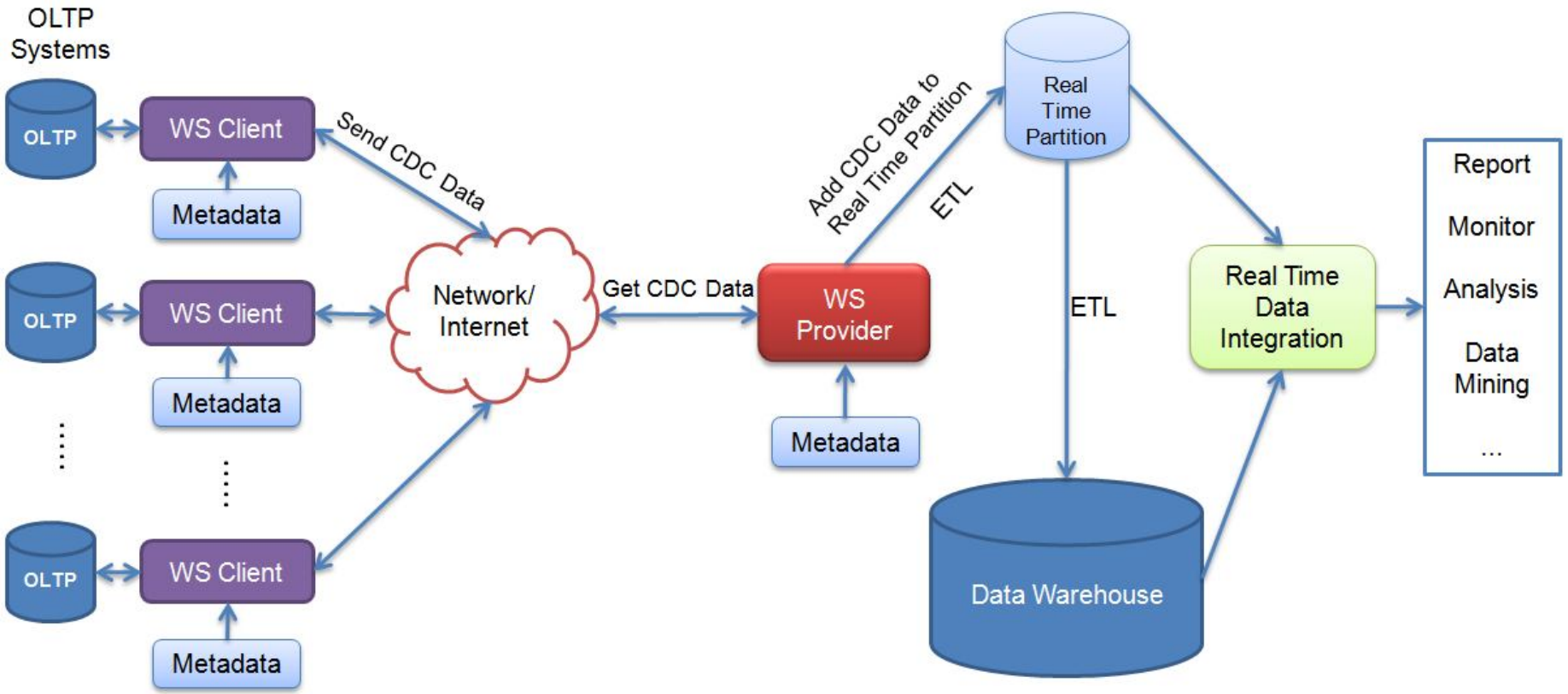

Figure 25: Web Services Based Real Time Data Warehouse Architecture

5.2.1 Web Service Client

This component is used for getting data changes (known as Change Data Capture - CDC) from OLTP systems and send data to the Web Service Provider by calling related web service. The real time data is captured through the trigger which inserts data an auxiliary table in the database. Our RTDW Web Service Client continuously polls the logs data and then pushes the data to the RTDW Web Service. Any data inserted, updated or deleted in an OLTP system sent to the RTDW Web Service component.

For determining tables which data is captured, RTDW Web Service Client uses OLTP metadata that is prepared for this purpose. The client uses an SQL-Generator to capture log data. The client calls the SQL-Generator with parameters system name, table owner and table name, then SQL-Generator returns the required SQL statement to capture log data. The client executes generated SQL statement and fetches data, and transfers data via Data Transfer Object which is a general object for structured table data. This object holds data and metadata about data which it contains.

In OLTP metadata, we also track the last-change-ids of data in the table. After getting data via RTDW Web Service Client, the data in the log table is deleted. The structure of this metadata table which holds data about tables and last log id is shown Table 4.

Table 4: Metadata Table Structure

Table Name: OLTP_TABLES		
Column Name	**Data Type**	**Size**
ID	NUMBER	
OWNER	VARCHAR2	30
TABLE_NAME	VARCHAR2	30
LOG_TABLE_NAME	VARCHAR2	30
LAST_LOG_ID	NUMBER	
PK_COLUMNS	VARCHAR2	255

5.2.1.1 Capturing the Data

Data that is required to data warehouse is collected from the operational systems and other external sources. Data capturing techniques in use mainly include; source data extraction, log capture, triggered capture, application-assisted capture, timestamp-based capture, and file comparison capture.

Source Data Extraction: This technique provides a static snapshot of source data at a specific point in time. Once the data warehouse database is constructed, all the data at this time have to be transferred from operational systems to DW. This operation is generally called ***initial load***, and after that starts capturing new changes.

Log Capture: Log capture is mainly used technique for collecting changes in source systems, but logging has to be supported and turned on the source system. The formats of the log records are also so important that it is clearly and easily understandable. Today, many commercial database systems have software solutions for log capture.

Triggered Capture: Database trigger is a code block that is executed when the predefined events (such as inserting, deleting or updating a row of a table) are occurred in the database. Many database management systems support triggers.

Application-assisted capture: The technique is writing programs to collect the data from the operational sources and completely under the control of programmer involving testing and maintenance responsibility.

Timestamp-based Capture: In this technique, timestamp values are used in the records. Timestamp values are used as a flag that indicates if the record has changed after the last capture or not.

File Comparison: At a specific point in time, a snapshot of the data source is taken and saved in a file and then it is compared with the previous snapshot file.

In our real time data warehouse architecture, we used triggered capture technique, since it can be implemented easily. Log capture might be chosen because of that its minimal overhead on source database system, but we have to use additional software and need additional configuration issues. Therefore, we used a log table for each table in our OLTP system and a log trigger on each source tables. Figure 26 shows this architecture.

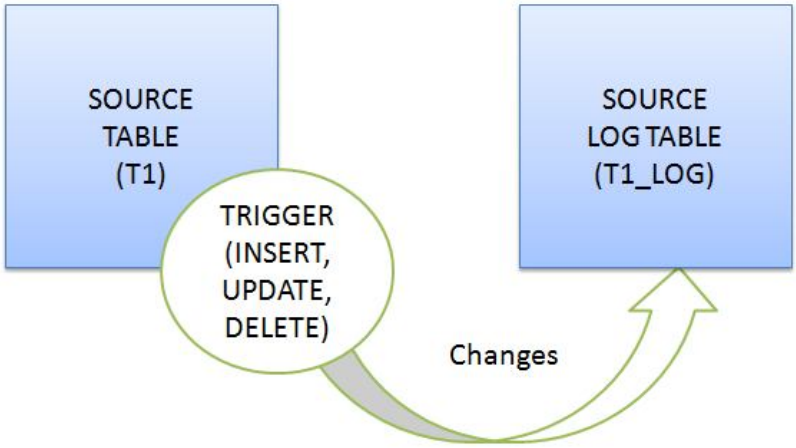

Figure 26: Log Capture Architecture on Source Tables

5.2.2 Web Service Provider

Web Service Provider is basically a web service which gets data sent by RTDW Web Service Client and adds to Real Time Partition.

This component gets Data Transfer Object which is sent by RTDW Web Service Client, decompose into two parts: data and metadata. RTDW Web Service uses metadata to generate SQL via SQL-Generator for inserting data to RTDW log tables then executes this generated SQL on RTDW database and inserts data.

5.2.3 Metadata

Metadata in this architecture is used for two purposes; first is defining and tracking OLTP tables from which changed data is captured, the second is defining warehouse tables and real time partition log tables. In this architecture, no database specific metadata is used. Therefore in this model, any database server can be used as an OLTP system and a data warehouse system.

5.2.4 ETL

ETL has extremely important role in data warehouse's establishment and the maintenance process. In this architecture, we used ETL in the warehouse part for transforming log data into aggregated data such as summations and counts.

5.2.5 Real Time Partition

Instant data changes (mostly daily) are put into this component first, then later merged into data warehouse. In this architecture, we used a novel structure

that is different from traditional real time partition implementations. In this structure we used three steps:

1. **Put CDC data into related warehouse log table:** In the warehouse part of the architecture we create log tables for each OLTP table from which data is captured. RTDW Web Service puts all the data sent by client into this log tables.

2. **Clean CDC log data on demand:** This step is firstly determines the latest state of the data in the log table and we use only latest data at this time for aggregations. We call this **clean delta**. For example, a data is first INSERTED and DELETED within the day, and at this moment, the latest status of data is DELETED and this step eliminates this data. After that, this step uses clean delta records to clean and format CDC data. Clean delta log type conversions are shown in Table 5.

Table 5: Clean Delta Log Type Conversions

First Record Log Type	Last Record Log Type	Result
INSERT	DELETE	ELEMINATE
INSERT	UPDATE	INSERT LAST RECORD
UPDATE	INSERT	UPDATE LAST RECORD
UPDATE	DELETE	DELETE LAST RECORD
DELETE	INSERT	UPDATE LAST RECORD
DELETE	UPDATE	UPDATE LAST RECORD

3. **Aggregate clean CDC data on demand:** In this architecture, also, the real time aggregations calculated on demand. If there is no request for real time data the aggregation does not calculated throughout the day. When data is requested from real time partition, only the related aggregations are calculated on demand. We call this **On Demand Aggregation.**

5.2.6 Data Warehouse

This component, as explained above chapters, not only holds the historical aggregated data but also today data via real time partition. Real time partition data puts into the data warehouse daily basis. When a new day begins, aggregation are calculated for yesterday data in the real time partition and put into data warehouse.

5.2.7 Real Time Data Integration

This component is used for integrating the data both in Real Time Partition and Data Warehouse. When a user sends a query to this component; if query only wants historical data then this component send the query to Data Warehouse, if query wants both historical and instant data then this component rewrites the query to get and integrate data.

In our case study, query rewriting is done by using views. First, we get the SQL from user and determine the date predicate. Then if date predicate is consist of today then we replace the fact table with our view which merges fact table and real time partition table as shown in Figure 27.

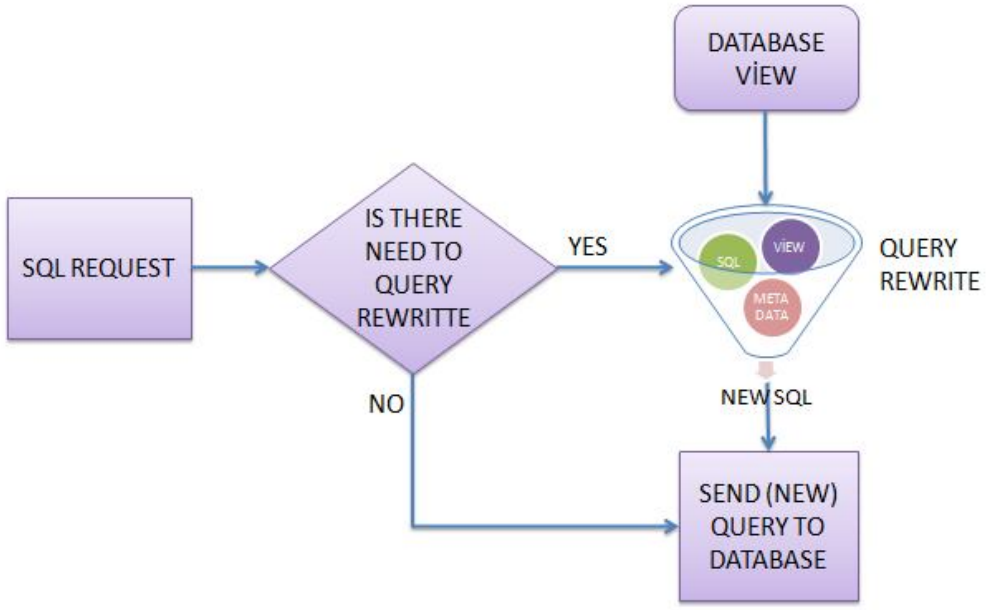

Figure 27: Query Rewrite Flow

5.3 Alternative Technologies

In this thesis study, before using web services, we tried another two different technologies which are Hazelcast (In memory data grid.) and Client Server.

Hazelcast is an open source clustering and data distribution platform for Java, which is:

- **Lightning-fast**; thousands of operations/sec.
- **Fail-safe**; no losing data after crashes.
- **Dynamically** scales as new servers added.
- **Super-easy** to use; include a single jar.

Hazelcast is in-memory data grid solution with its various distributed data structures, distributed caching capabilities, elastic nature, memcache support, integration with Spring and Hibernate. Hazelcast is released under Apache License and the project is hosted at Github. It can be freely used in commercial or non-commercial applications [53].

The **client/server** model is a distributed application model which partitions tasks or workloads between servers and clients. Clients and servers communicate over a computer network. A server machine is a host that is running one or more server programs. Server programs share their resources with clients. A client requests a server's content or service function. Therefore Clients initiate communication sessions with servers which await incoming requests [54].

In this study, first we tried Hazelcast. Since it's a grid solution, it sends data which is captured from one client to all machines (clients and servers). We tried to direct to the Data Warehouse machine, but it did not allowed this. Solutions to this problem is against to Hazelcast architecture and not easy to use, because it has developed for a grid solution.

In addition, we tried Client-Server solutions that we implemented a client-server program by using sockets. However, this solution is good for transferring text data, but it needs too many controls and also a data transfer protocol for transferring objects. There are some solutions to transfer objects but they are not easily usable.

As a result, we have chosen to use web services to transfer data from source machines to target server machine. Web services are easy to implement and we could develop a solution very fast.

A basic comparison of these technologies is shown in Table 6. From the table we say that the web services based architecture can be chosen for RTDW implementations.

Table 6: A Basic Comparision of Technologies

Properties	Hazelcast Based	Client-Server Based	Web Services Based
Use of http protocol	No	No	Yes
Dynamically scales	Yes	Development Specific	Yes
Redundant	Yes	No	No
Targetted to one server	No	Yes	Yes
Object transfer is easily implemented	Yes	No	Yes
Grid solution	Yes	No	No
Easy to implement	Yes	No	Yes
Raw data transfer	Yes	No	No

5.4 Similar Solutions

In this section, we briefly described some today's solutions proposed some of the companies. The solutions are new in the market and there are not many case studies about them.

5.4.1 Oracle Data Integrator and GoldenGate

Conventional ETL tools closely intermix data transformation rules with integration process procedures, requiring the development of both data transformations and data flow. Oracle Data Integrator (ODI) takes a different approach to integration by clearly separating the declarative rules (the "what") from the actual implementation (the "how"). This approach for declarative design has also been applied to ODI's framework for Changed Data Capture. ODI's CDC moves only changed data to the target systems and can be integrated with Oracle

GoldenGate, thereby enabling the kind of real time integration that businesses require [55].

In order to provide no to low latency loads, ODI has alternative solutions for real-time data warehousing through the use of CDC mechanism, including the integration with Oracle GoldenGate. This integration also provides seamless operational reporting. Data federation and data service use cases are covered by Oracle Data Service Integrator (ODSI) [55].

5.4.1.1 Methods for Tracking Changes using CDC

Oracle generally uses three different techniques for capturing CDC data [55].

- Trigger Based
- Streams Based
- GoldenGate Based

In the first technique, database triggers are defined that are executed inside the source database when a table change occurs as shown in Figure 28.

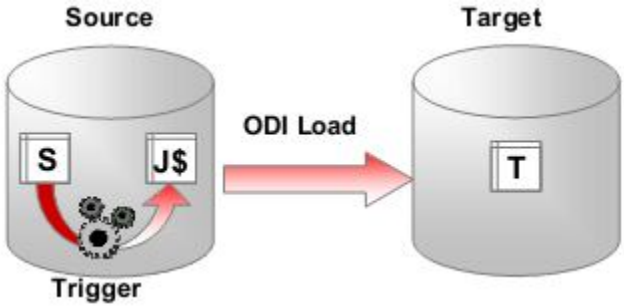

Figure 28: ODI Trigger Based Capture [55]

Some databases provide APIs and utilities to process table changes programmatically. Oracle database provides the Streams interface to process log entries and store them in separate tables. ODI also supports log-based CDC. This is shown on Figure 29.

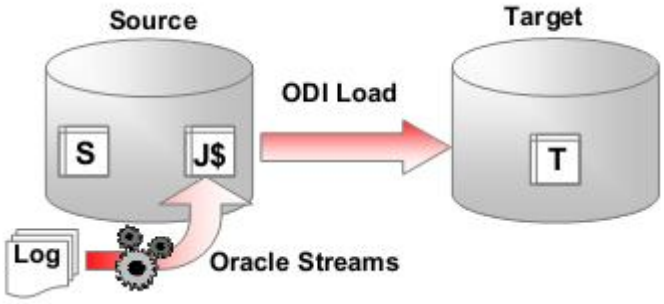

Figure 29: Streams Based CDC [55]

Oracle GoldenGate provides a CDC mechanism that can process source changes by processing log files of completed transactions and storing these captured changes into external Trail Files independent of the database. Changes are then transferred to a staging database. These changes will be loaded into the target data warehouse using ODI's declarative transformation mappings. This architecture enables separate real-time reporting on the normalized staging area tables in addition to loading and transforming the data into the analytical data warehouse tables. This is shown in Figure 30.

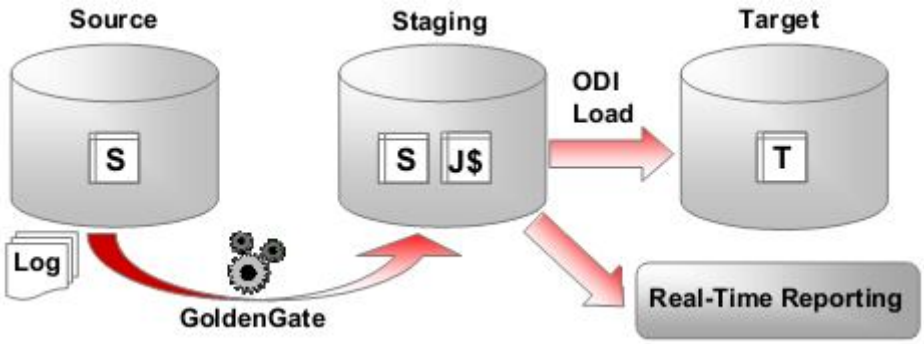

Figure 30: GoldenGate Based CDC [55]

ODI processes datastore changes in two ways [55]:

- **Regularly in batches (pull mode):** For example, processes new orders from the Web site every five minutes and loads them into the operational datastore (ODS).
- **In real time (push mode) as the changes occur:** For example, when a product is changed in the enterprise resource planning (ERP) system, immediately updates the on-line catalog.

In practice, for Oracle, there is one approach that satisfies the majority of real-time data warehousing use cases: **The micro-batch approach using**

GoldenGate-based CDC with ODI. This is shown in Figure 31. In this approach, one or more tables from operational databases are used as sources for GoldenGate CDC into a staging area database. This staging area provides a real-time copy of the transactional data for real-time reporting using BI tools and dashboards. The separate staging area handles operational BI queries without adding load to the transactional system. ODI performs a load of the changed records to the real-time data warehouse in frequent periods of 15 minutes or more [55].

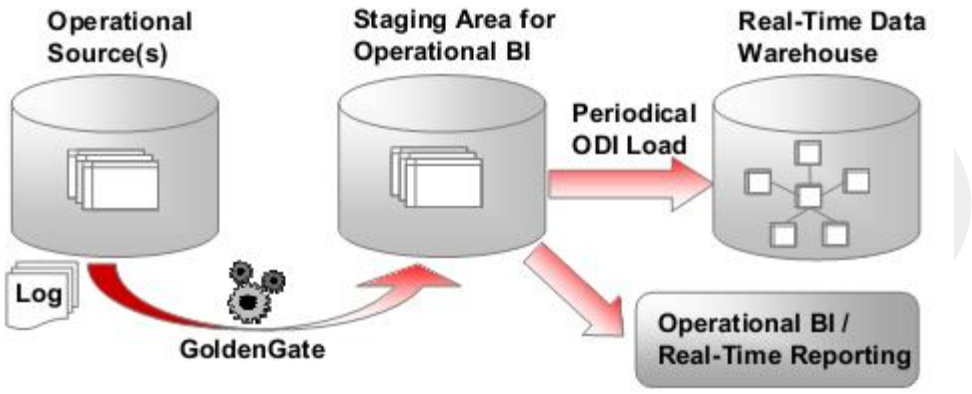

Figure 31: Micro-Batch Architecture using ODI and GoldenGate [55]

5.4.2 SQLStream

SQLstream enables multiple sources of heterogeneous data to be aggregated, correlated and filtered in real-time. Change Data Capture adapters provide the real-time 'Extract' function, delivering a stream of relational data from the source systems. SQLstream processes the relational data streams, providing both the 'Transform' and 'Aggregation' functions in a single platform. Most importantly, the data is aggregated in SQLstream, in real-time, before the 'Load' operation takes place. Aggregate and raw data are pushed from SQLstream into the data warehouse in real-time [56].

5.4.3 iWay Data Integration

iWay Software data integration solution allow for direct access to data. iWay supports extract, transform, and load; enterprise information integration initiatives; and web services deployments [57]. The iWay CDC solution is shown in Figure 32.

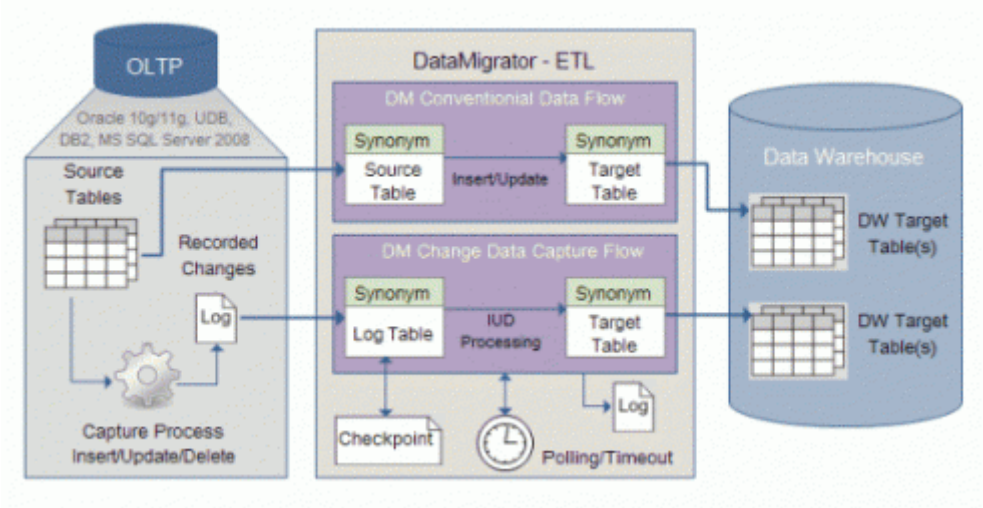

Figure 32: iWay CDC Solution [57]

Service Manager provides real-time event listening capabilities for sources including applications, databases, cloud, and legacy systems. Each event can trigger a variety of business processes while feeding the data warehouse in real-time [57].

DataMigrator Change Data Capture provides a real-time capability using database logs to read only the changes (inserts, updates, and deletions) made to tables in any of the major relational databases and delivers those changes to DataMigrator. DataMigrator CDC makes database logs from disparate databases available in a common format so that they can be used as a data source. Processing the changes as they occur lets a data warehouse provide near real-time access to operational data. A polling interval (how often to check for changes) can be specified as often as once a minute. A timeout interval (how long to keep checking) can also be specified. Once a CDC source is configured, it can be read using SQL and used like other data sources [57].

5.4.4 Microsoft StreamInsight

Microsoft StreamInsight is a platform to build low-latency event-driven analytics applications. StreamInsight is available as part of Microsoft SQL Server since Microsoft SQL Server 2008 R2 in April 2010. StreamInsight complements SQL Server with new capabilities to build event-driven solutions and to inject time-based analytics into the event processing pipeline. This enables organizations to be event-driven: analytical results are available for human consumption right away, or systems can react to events independently based on automated

workflows [58]. Figure 33 depicts the developer and runtime experience of a StreamInsight application and introduces some of the key concepts.

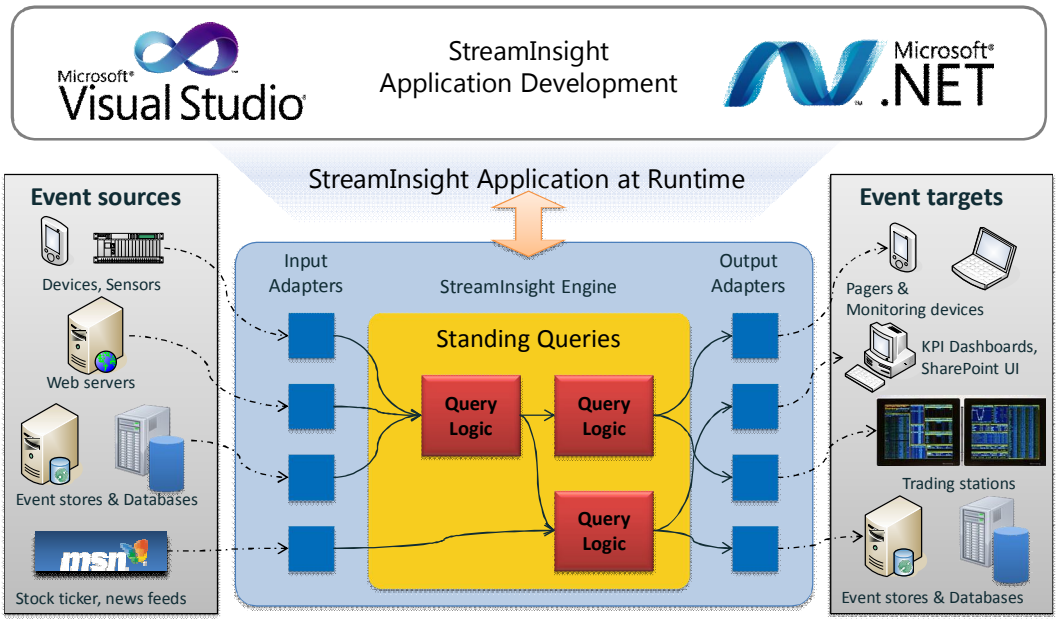

Figure 33: StreamInsight Application Development and Runtime

CHAPTER 6

A CASE STUDY

This section describes a case study of implementing a RTDW for an **Inventory Control System**. Inventory control systems are used for managing the stocks of companies and big distribution organizations. Therefore, it's valuable to build a RTDW for decisions which needs historical and current status.

In this study, we designed a database for simulating an OLTP application. The aim of this case study is to build a RTDW to enable analysis of the data in the OLTP database. The conceptual design is modeled using Kimball model. And finally this case study illustrates an implementation of a real time data warehousing solution covering all phases of design. We have chosen Oracle 11g R2 as database server for OLTP and RTDW systems, since it's widely used database in the world.

6.1 Modeling the Database

The first step of building (or creating) a database after getting business requirements is to construct a data model, which represents the business process, its attributes and relationships between them. In our case study we constructed two models: one for OLTP and one for OLAP.

6.1.1 Entity Relation (E-R) Model

Entity-relationship (E-R) modeling is a high-level data modeling technique that is originally developed by Professor Peter Chen to serve as a tool for communication between designers and users. E-R models are best expressed using graphical E-R diagrams.

An E-R data model is a high-level conceptual model that describes data as entities, attributes, and relationships and pays particular attention to the interactions among entities. In the development of databases, relationships are the glue that holds information together and their realization in relational databases is particularly important [59].

In our case study, we modeled a database for simulating an OLTP application. Figure 34 shows the E-R model diagram of this OLTP database.

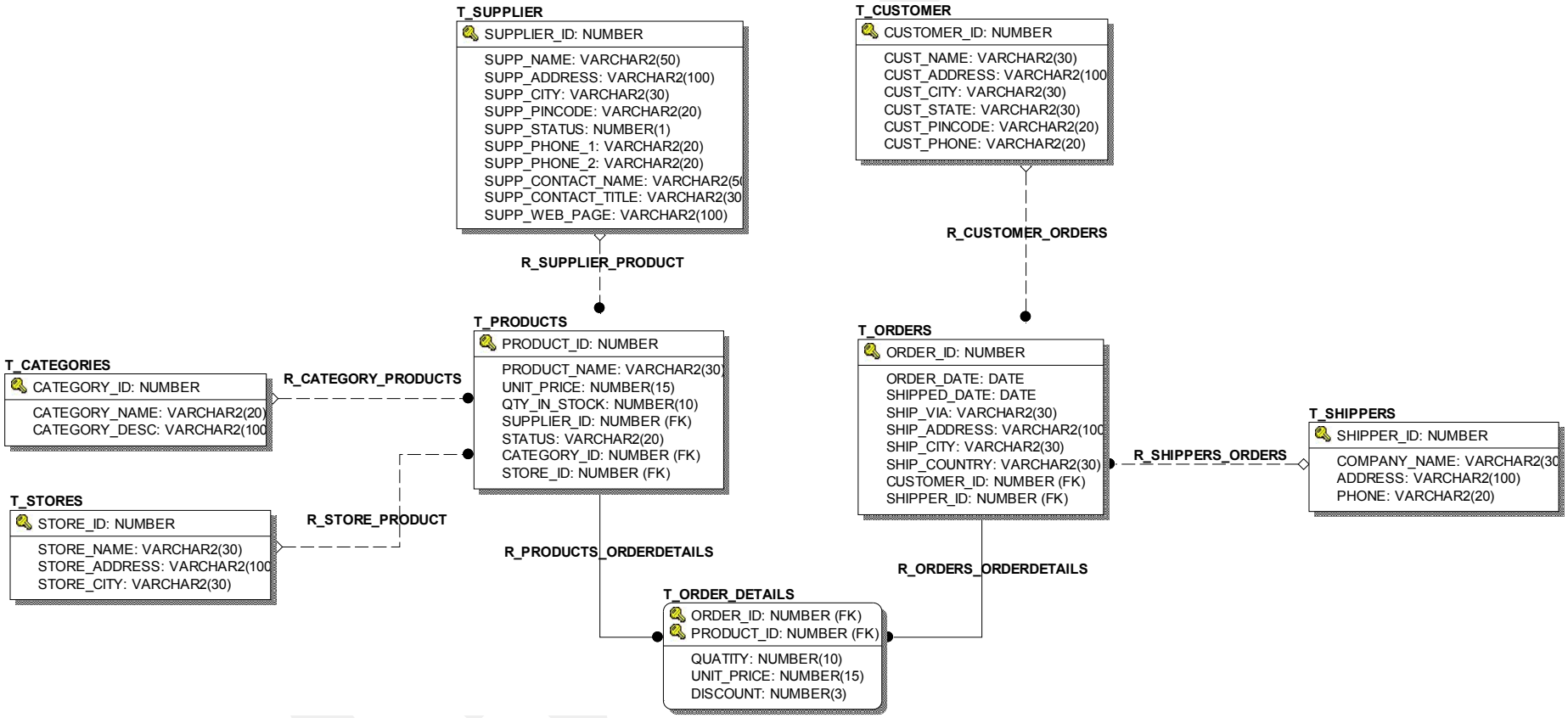

Figure 34: E-R Model Diagram of Sales Schema

6.1.2 Dimensional Modeling

Dimensional modeling (DM) is a logical design technique that tries to present the data model in an accessible and intuitive standard frame and favorite in data warehousing. This data is visualized as a set of measures that are defined according to the business. The purpose is optimizing decision support query performance in relational databases, relative to a measurement or set of measurements of the outcome(s) of the business process being modeled [60].

In our case study, we modeled a database for OLAP purposes. Figure 35 shows the dimensional model diagram of this OLAP database.

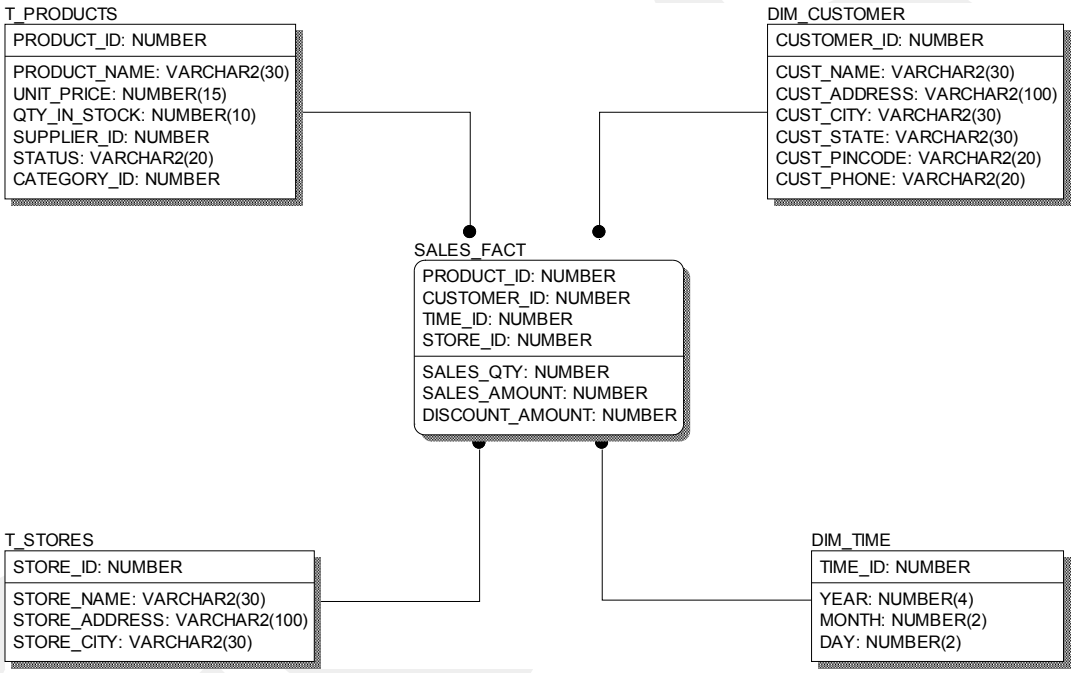

Figure 35: Dimensional Model Diagram of Sales Schema

6.2 Data Loading and Transformation

Design of the data loading and transformation strategy is another crucial point. General data loading processes which we used in this study is shown in Figure 36. In our case study, we examined and discussed the strategies which we would decide to use.

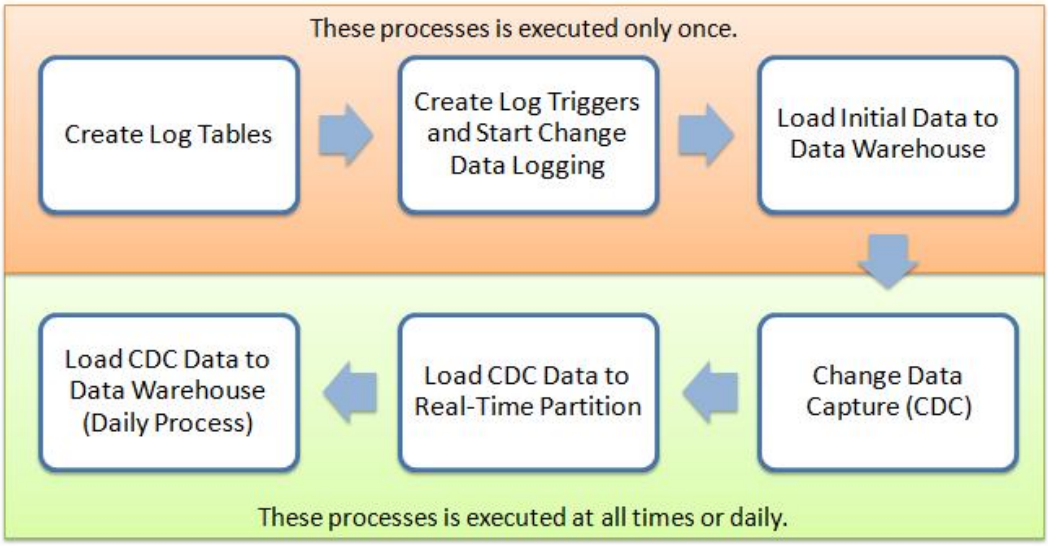

Figure 36: General Data Loading Processes

6.2.1 Capturing the Data

In this study, we used trigger to capture changed data. For this purpose we created log triggers and log tables for every table which we track changes in our OLTP system. This technique can be implemented easily implemented since many databases support this feature. Log capture might be chosen because of that its minimal overhead on source database system, but we have to use additional software and need additional configuration issues. Therefore, we created a log table for each table in our OLTP system and a log trigger on each source tables. The OLTP system begins to log changes immediately after log tables and log triggers created.

6.2.2 Transforming the Data

The data transformation process is converting the captured data into a format and structure suitable for loading into the data warehouse. Data transformations are often the most complex part of the ETL process. They can range from simple data conversions to extremely complex data scrubbing techniques. From an architectural perspective, there are two ways to transform data: [61]

- Multistage Data Transformation
- Pipelined Data Transformation

The Multistage Data Transformation logic for most data warehouses consists of multiple steps. For example, to insert new records into a sales table, there may be separate logical transformation steps to validate each dimension key.

In Pipelined Data Transformation, the ETL process flow can be changed dramatically and the database becomes an integral part of the ETL solution. The new functionality shifts from serial transform-then-load process (with most of the tasks done outside the database) or load-then-transform process, to an enhanced transform-while-loading.

The transformation process includes standardizing, integrating, cleansing, augmenting, aggregating and creating the data sets for loading into the repository. The main types of ETL transformations done in the data are as follows: [62]

- Creating common keys:
- Creating surrogate keys
- Standardizing the descriptions, textual attributes:
- Translation and standardization across organization standards & structures
- Transformation for common dimension attributes
- Data Quality
- Data Relevance
- De-Duping, Merging and data cleansing
- Data Augmentation and enrichment
- Data Type conversion
- De-normalization
- Normalization
- Create Derived Attributes
- Calculation, Derivative, Allocation
- Aggregation

In our case study, we used PL/SQL for data transformation. For example, aggregations such as sales quantity, sales amount, is calculated by using PL/SQL. Figure 37 shows data transformation architecture that we used.

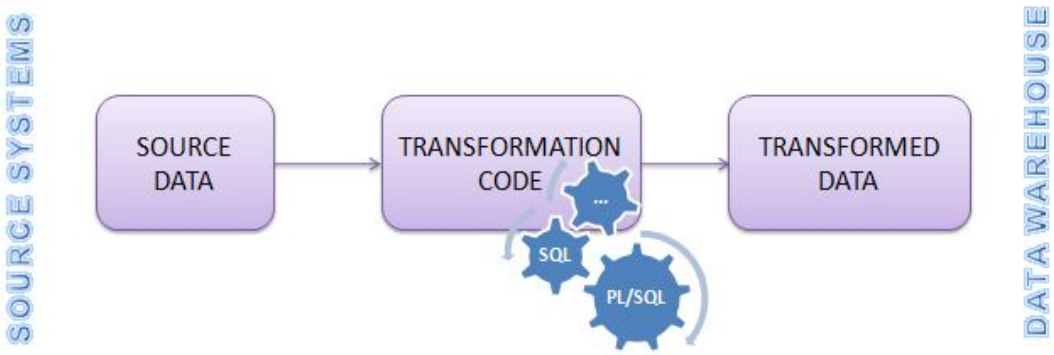

Figure 37: Data Transformation Architecture

6.3 Real Time Data Integration

Real Time Data Integration is used for integrating the data both in Real Time Partition and Data Warehouse. When a user sends a query to this component; if query only wants historical data then this component send the query to Data Warehouse, if query wants both historical and instant data then this component rewrites the query to get and integrate data.

In our case study, query rewriting is done by using views. First, we get the SQL from user and determine the date predicate. Then if date predicate is consist of today then we replace the fact table with our view which merges fact table and real time partition table.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Real time data warehouse is much more than a new feature. Moving to real time delivery of data challenges every aspect of the data warehouse.

We designed, developed and built a web services based real time data warehouse. Web services are important component as they communicate easily to the servers which publish services. Because of easily adaptable and maintains many of the communication problems, we think that web services are preferable choice.

Capturing change data from source systems is also a major problem for data warehouse constructions. Log capture and triggering is mostly used techniques in this area. We think log capture may be preferred to triggering which gets come overhead to the source system database. However, it's easy to implement and need nothing except source database. Since today, triggers are supported most of the databases, it may be a good choice.

Real time partition is another important issue for building a real time data warehouse. Because of the design purposed of the data warehouses, you could not load instant data to your data warehouse immediately. You need a staging table before aggregating data and load it to the data warehouse, because your data is completed later (for example at the end of the day). Therefore, we use real time partition. In addition to that, if user requests real time data for his analysis then the data in the real time partition and data warehouse have to be merged. For this purpose, we used query rewriting which decides to rewrite and replaces the fact table with a view joined by real time partition.

7.1 Future Work

As a future work different change data capture methods can be applied to data capturing process. Unstructured data issues in this area can also be researched.

Additionally new approaches can be applied to manage the real time partition. One approach may be using an in-memory database or a caching mechanism to hold real partition time data, but at this time new data integration issues arises, because of the using different databases.

Finally many query re-writing methods can be implemented to integrate real time partition and data warehouse data.

REFERENCES

[1] ELMASRI, R.A., NAVATHE S.B. (2010), Fundamentals of Database Systems.

[2] ULLMAN, J., WIDOM, J., GARCIA-MOLINA, H. (2008), Database Systems: The Complete Book.

[3] SIBERCHATZ, A., KORTH, H.F., SUDARSHAN, S. (2010) Database System Concepts.

[4] NEBIKER, S., BLEISCH, S. (2010) Relational Database Design and Implementation

[5] HARRINGTON, J. (2009) Database Systems:Concepts and Architectures, Report, Geographic Information Technology Training Alliance (GITTA)

[6] WWW.DATAWAREHOUSE4U.INFO (2009), Internet: http://datawarehouse4u.info/OLTP-vs-OLAP.html, [Nov., 18, 2011].

[7] ARIAS E. (2011) OLTP vs OLAP - Definition and Differences, Internet: http://axwonders.blogspot.com/2011/12/oltp-vs-olap-definition-and-differences.html, [May, 15, 2012].

[8] RAINMAKER, Rainmaker Data Warehousing, Internet: http://www.rainmakerworks.com/whitepapers.html, [Nov., 18, 2011].

[9] WARD, P., DAFOULAS,G. (2006), Database Management Systems

[10] DYCHE, J. (2000), e-Data: Turning Data into Information with Data Warehousing.

[11] WWW.DATAWAREHOUSE4U.INFO (2008), Internet: http://datawarehouse4u.info/index_en.html, [June, 21, 2012].

[12] INMON W.H. (2005), Building the Data Warehouse.

[13] KIMBALL R. (2002), The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.

[14] HAN, J., KAMBER, M. (2011), Data Mining: Concepts and Techniques.

[15] JARKE, M. et. Al (2003), Fundamentals of Data Warehouses.

[16] GUANGQUAN. Z., YANG. X., TIANRUI, L. (2011), Knowledge-Based Systems, A special issue on new trends in Intelligent Decision Support Systems

[17] LIAUTAUD, B., HAMMOND, M. (2000), E-Business Intelligence: Turning Information into Knowledge into Profit.

[18] UTLEY, C., (2009) Designing the Star Schema, Internet: http://www.ciobriefings.com/Publications/WhitePapers/DesigningtheStarSchemaDatabase/tabid/101/Default.aspx, [Jan. 05, 2012].

[19] KIMBALL, R., CACERTA, J. (2004), The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data.

[20] WWW.OLAP.COM, Internet: http://olap.com/w/index.php/Drill_Up, [Jan. 23, 2012].

[21] JORG, T., DESSLOCH, S. (2009), Formalizing ETL Jobs for Incremental Loading of Data Warehouses,GI-Fachtagung fur Datenbanksysteme in Business, Technologie und Web, Lecture Notes in Informatics.

[22] TECH TARGET: SEARCH SQL SERVER, Internet: http://searchsqlserver.techtarget.com/definition/pivot-table, [Feb. 02, 2012].

[23] GEEKINTERVIEW, Internet: http://www.geekinterview.com/question_details/27825, [Feb. 17, 2012].

[24] WAH, T.Y., PENG, N.H., HOK, C.S. (2007), Building Data Warehouse, 24th South East Asia Regional Computer Conference

[25] KIMBALL, R. (2005), Internet: http://www.kimballgroup.com/html/designtipsPDF/DesignTips2005/DTKU63BuildingAChangeDataCaptureSystem.pdf, [Feb. 22, 2012].

[26] KIMBALL, R., ROSS M. (2010), The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence.

[27] HAAS, L.M. et al. (1999), Transforming Heterogeneous Data with Database Middleware: Beyond Integration, IEEE Data Eng. Bull.

[28] DATA-WAREHOUSES.NET, Internet: http://data-warehouses.net/architecture/staging.html, [June 12, 2012].

[29] INMON, B., IMHORF, C., BATTAS, G. (1998), Building The Operational Data Store.

[30] BARAGOIN, C. et al. (2001), Building the Operational Data Store on DB2 UDB.

[31] MUNOZ, L., MAZON, J.N., TRUJILO, J. (2011), ETL Process Modeling Conceptual for Data Warehouses: A Systematic Mapping Study, IEEE Latin America Transactions, Vol. 9, No. 3.

[32] LUMPKIN, G. (2009), Oracle Database 11g for Data Warehousing and Business Intelligence.

[33] BRESLIN, M. (2004), Data Warehousing Battle of Giants: Comparing the

Basics of Kimball and Inmon Models, Business Intelligence Journal.

[34] INMON, W.H., IMHORFF, C. (2007) Internet: http://www.inmoncif.com/library/cif/, [Mar. 11, 2012].

[35] INMON, W.H. (2002), Building the Data Warehouse.

[36] PAUL, L. (2007), Oracle Database Data Warehousing Guide 11g Release 1.

[37] EXECUTION-MIH (2012), Internet: http://www.executionmih.com/data-warehouse/slowly-changing-dimension-SCD.php, [May 09, 2012].

[38] MUNDY, J. (2007), Kimball University: Handling Arbitrary Restatements of History.

[39] VASSILIADIS, P., SIMITSIS, A. (2009), Near Real Time ETL: New Trends in Data Warehousing and Data Analysis. Springer Science+Business Mediapp.

[40] LANGSETH, J. (2008), Decision Support Systems Resources, Internet: http://dssresources.com, [May 18, 2012].

[41] VUSOLUTIONS (2012), Internet: http://vusolutions.com/tag/vu-current-solved-gdbs-bba/, [May 23, 2012].

[42] JOVANKA, A., VALTER, F. (2003), Design and Management of Data Warehouses (DMDW), Data Warehouse Population Platform.

[43] CHAPPELL, D.A. (2004), Enterprise Service Bus.

[44] NAEEM, M.A., DOBBIE, G., WEBER, G. (2008), An Event-Based Near Real-Time Data Integration Architecture, 2008 12th Enterprise Distributed Object Computing Conference Workshops.

[45] GUERRA, J., ANDREWS, D.A. (2011), Creating a Real Time Data Warehouse

[46] HAISTEN, M. (2000), Real-Time Data Warehouse: What is Real Time about Real-Time Data?, DM Review.

[47] MADSEN M. (2008), How Real Time Data Requirements Change the Data Warehouse Environment.

[48] RUSSOM, P. (2011), Big Data Analytics, TDWI best practices report.

[49] GATHIBANDHE H., DEOGIRIKAR S., GUPTA A.K. (2010), Information Management, Internet: http://www.information-management.com/infodirect/2009_152/real_time_business_intelligence-10017057-1.html?zkPrintable=1&nopagination=1, [June 8, 2012].

[50] GABLE, JULIE (2002), Enterprise Application Integration.

[51] ZHONG L. (2004), Application of the Web Service Technology on the Data Warehouse System, Jounal of WuHan University of Technology Vol.26 No.8.

[52] ANKORION, I. (2005), Change Data Capture-Efficient ETL for Real-Time BI, DM Review Magazine.

[53] HAZEL (2011), The Art of Data Distribution, Internet: www.hazelcast.com, [Feb. 02, 2012].

[54] REESE, G. (2000), Database Programming with JDBC & Java.

[55] ORACLE (2010), Best Practices for Real-time Data Warehousing.

[56] SQLSTREAM (2012), Transform Big Data into Real Time Value.

[57] INFORMATION BUILDERS (2012), Real-Time Data Warehousing, Internet:http://www.informationbuilders.com/products/integration/data/realti

me, [July 28, 2012].

[58] TOPP, M., APOSTOKALIS, I., GRABS, T. (2012), Next Generation Energy and Manufacturing Analytics. Internet:http://www.informationbuilders.com/products/integration/data/realtime, [July 28, 2012].

[59] RICHARDI, G. (2003), Database Management with Web Site Development Applications.

[60] KIMBALL, R. (1997), A Dimensional Model Manifesto, Internet: http://www.kimballgroup.com/html/articles_search/articles1997/9708d15.html, [Jun. 05, 2012].

[61] ORACLE (2005), Oracle® Database Data Warehousing Guide 10g Release 2 (10.2) Internet: http://docs.oracle.com/cd/B19306_01/server.102/b14223/transform.htm, [May 28, 2012].

[62] EXECUTION-MIH (2012), Internet: http://www.executionmih.com/data-warehouse/etl-transformation-design.php, [May 09, 2012].

APPENDIX A

CIRRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Obalı, Murat

Nationality: Turkish (T.C.)

Date and Place of Birth: 10 March 1976, Konya

Marital Status: Married

Phone: +90 555 460 34 59

Email: muratobali@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
MS	Çankaya Univ. Computer Engineering	2012
MS	Niğde Univ. Business Administration	2002
BS	Ege Univ. Computer Engineering	1999
High School	Fatih Anatolia High School	1994

WORK EXPERIENCE

Year	Place	Enrollment
2011 - Present	TUBITAK	Senior Researcher

2004 - 2011	HAVELSAN A.Ş.	Team Leader
2001 - 2003	UNDP	Information Systems Consultant
2000 - 2003	METEKSAN SISTEM A.Ş.	Project Manager
1999 - 2000	BILIŞIM LTD.	Expert Programmer