

# **IMPORTANCE OF METADATA IN DATA WAREHOUSING**

---

A Thesis

Presented to the

Faculty of

San Diego State University

---

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Abhinav Dhiman

Summer 2012

**SAN DIEGO STATE UNIVERSITY**

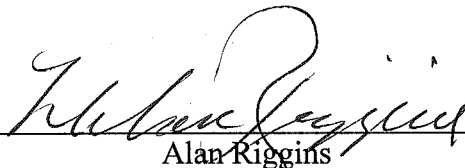
The Undersigned Faculty Committee Approves the

Thesis of Abhinav Dhiman:

Importance of Metadata in Data Warehousing



Joseph Lewis, Chair  
Department of Computer Science



Alan Riggins  
Department of Computer Science



Michael E. O'Sullivan  
Department of Mathematics and Statistics



Approval Date

Copyright © 2012

by

Abhinav Dhiman

All Rights Reserved

## **DEDICATION**

This thesis is dedicated to my mother and father, who taught me to love and respect for knowledge and big tasks can only be achieved by achieving one step at a time.

## **ABSTRACT OF THE THESIS**

Importance of Metadata in Data Warehousing

by

Abhinav Dhiman

Master of Science in Computer Science

San Diego State University, 2012

Data warehouses are an integral component for business decision-making when a large volume of data has to be analyzed. Data in data warehouses come from sources including, but not limited to relational databases, legacy systems, and flat files. The data from these sources have to be processed to conform to the data model and type of the data warehouse. Extract, Transform, and Load (ETL) tools are used to take data from multiple sources to populate a data warehouse, typically into a star schema. After the data is loaded into the star schema, it can be used for query from reporting tools, also known as the reporting layer of data warehouse.

There are three parts to the data warehousing, first is the source systems, second is the ETL processing, and third is the reporting part. In order for well functioning of all the parts, they need to work together comprehensively. Metadata, usually called data about the data is a vital component necessary in to make sure the smooth working of data warehouse. Metadata ties all the loose ends of the total parts involved in data warehouse. Metadata has to be generated at all the levels and stages of data warehouse.

The main purpose of this research is to bring up the importance of modeling of metadata model while designing data warehouse. This thesis presents the metadata model for source metadata and for ETL metadata that can be applied to any data warehouse. In most of the design scenarios the metadata is not focused during the development phase of data warehouse. According to Ralph Kimball, there is no standard way of doing the metadata implementation in data warehouse. By presenting a metadata model, the main aim to understand the importance of metadata modeling and a standard way of implementing the metadata in data warehouse.

## TABLE OF CONTENTS

	PAGE
ABSTRACT.....	v
LIST OF FIGURES .....	viii
ACKNOWLEDGEMENTS.....	x
CHAPTER	
1 INTRODUCTION .....	1
1.1 Motivation Example.....	4
1.2 Objective of the Research .....	5
1.3 Research Methodology .....	5
1.4 Overview of the Thesis .....	5
2 INTRODUCTION TO DATA WAREHOUSING AND METADATA .....	7
2.1 What Data Warehouse Really Is and Isn't?.....	7
2.2 OLTP and OLAP .....	9
2.3 Architecture of Data Warehouse.....	10
2.3.1 The Back Room .....	10
2.3.2 The Front Room: Data Access .....	13
2.3.3 OLAP .....	14
2.4 Data Warehouse Modeling .....	14
2.4.1 Conceptual Data Model .....	14
2.4.2 Logical Data Model .....	15
2.4.3 Physical Data Model .....	16
2.4.4 Dimensional Data Model .....	17
2.5 Data Warehouse Metadata .....	19
2.5.1 Source System Metadata.....	22
2.5.2 Back Room Metadata.....	22
2.5.3 Front Room Metadata .....	23
3 METADATA METHODOLOGY OF DATA WAREHOUSING .....	24
3.1 Introduction.....	24

3.2 Metadata Model for Data Warehousing.....	24
3.3 Metadata Model Details .....	25
3.3.1 Source Metadata Model .....	25
3.3.2 ETL Job Metadata Model .....	28
3.3.2.1 ETL_Job_Setup.....	28
3.3.2.2 ETL_Subjob_Setup.....	29
3.3.2.3 ETL_JobRun_Info .....	30
3.3.2.4 ETL_SubJobRun_Info .....	31
3.3.2.5 ETL_Job_Dependency.....	32
3.3.2.6 ETL_JobSoureDestination_TableInfo .....	32
3.4 ETL Job Framework Model and Algorithm .....	33
3.5 Explanation of the Algorithm .....	35
4 IMPLEMENTATION AND TESTS .....	37
4.1 Introduction.....	37
4.2 Implementation Platform .....	37
4.3 Test and Results .....	37
4.3.1 Source Metadata Implementation .....	37
4.3.2 ETL Metadata Implementation .....	41
4.4 Thesis High Level Summary .....	45
5 RESEARCH CONTRIBUTIONS AND POSSIBLE FUTURE WORK .....	49
5.1 Research Contributions .....	49
5.2 Future Work .....	50
REFERENCES .....	51

## LIST OF FIGURES

	PAGE
Figure 1.1. A data warehouse model. ....	1
Figure 2.1. An overview of OLTP and OLAP system.....	9
Figure 2.2. A typical data warehouse architecture.....	11
Figure 2.3. The four back room steps of a data warehouse (ETL). ....	12
Figure 2.4. An example of multidimensional data cube. ....	15
Figure 2.5. Example of conceptual data model.....	16
Figure 2.6. Example of logical data model. ....	17
Figure 2.7. Example of physical data model.....	18
Figure 2.8. An example of star schema.....	19
Figure 2.9. Example of snow flake schema. ....	20
Figure 2.10. Inmon metadata model.. ....	21
Figure 2.11. Data warehouse metadata model approach. ....	21
Figure 3.1 Data warehouse metadata model (high level thesis approach).....	25
Figure 3.2. Source metadata model.....	27
Figure 3.3. ETL job metadata model. ....	29
Figure 4.1. Results #1. ....	38
Figure 4.2. Results #2. ....	39
Figure 4.3. Results #3. ....	40
Figure 4.4. Results #4. ....	40
Figure 4.5. Results #5. ....	40
Figure 4.6. Results #6. ....	41
Figure 4.7. Results #7. ....	42
Figure 4.8. Results #8. ....	42
Figure 4.9. Results #9. ....	43
Figure 4.10. Results #10. ....	43
Figure 4.11. Results #11. ....	44
Figure 4.12. Results #12. ....	44



Figure 4.13. Results #13. ....	45
Figure 4.14. Start schema example. ....	47

## **ACKNOWLEDGEMENTS**

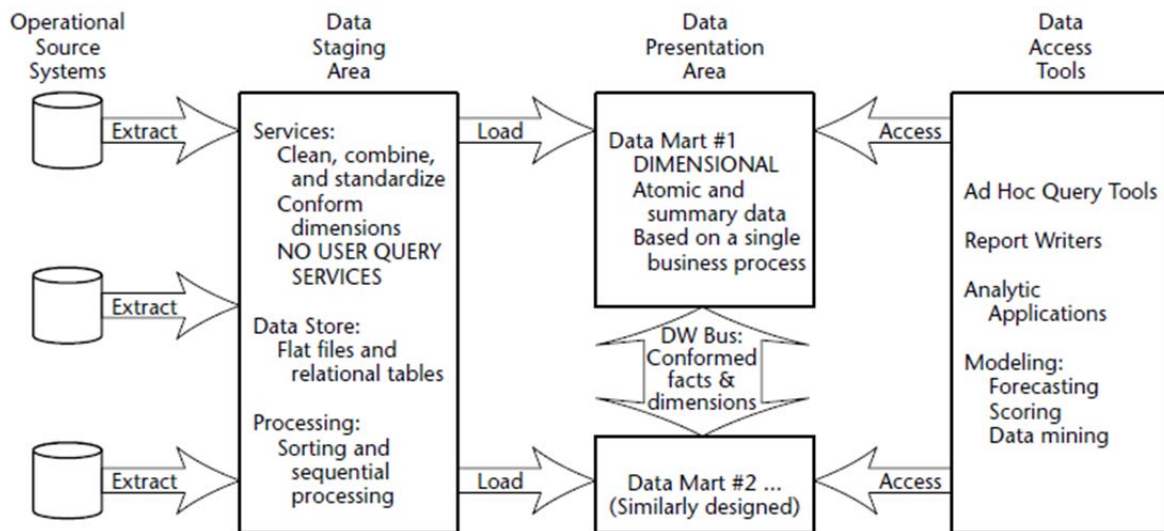
Prof. Joseph Lewis has been the ideal thesis supervisor. His advice, insightful criticisms, and patient encouragement gave me the support to write this thesis. I would also like to thank Prof. Alan Riggins and Prof. Michael E. O'Sullivan who showed faith in my ability to write this thesis.

## CHAPTER 1

### INTRODUCTION

A data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making. In data warehouse the data is stored in the form that can support decision making and analysis and are called as Online Analytical Processing Systems (OLAP) [1]. The OLAP systems are different from traditional OLTP systems, known as Online Transactional Processing Systems. OLTP systems are highly normalized data model to make the performance of the queries faster but in OLAP the data model is based on denormalization to make aggregated data that is used in analysis and decision making for business. [1]

According to Ralph Kimball, the data warehouse consists of four main components [2]. Please refer to the Figure 1.1.



**Figure 1.1. A data warehouse model.**

*Operational Source Systems:* These are operational systems of record that capture the transactions of the business. The source systems should be thought of as outside the data warehouse because we have little or no control over the data in these operational systems. [2]

*Data Staging Area:* The data staging area in data warehouse is both a storage area and a set of processes commonly referred to as *extract-transformation-load* (ETL) [2]. The data staging area is everything between the operational source systems and the data presentation area. Extraction is the first step in the process of getting data into the data warehouse environment. Extracting means reading and understanding the source data and copying the data needed for the data warehouse into the staging area. Once the data is extracted to the staging area, there are potential transformations, such as cleansing the data (standardizing the data format, dealing with special characters in data, resolving domain conflicts), combining data from multiple sources, and assigning warehouse keys. These transformations are necessary to load the data into the data warehouse presentation area. [3]

*Data Presentation Area:* The data presentation area is where data is organized, stored, and made available for direct querying by users, report writers, and other analytical applications.

*Data Access Tools:* The final component of the data warehouse environment is the *data access tools* [2]. The term tool refers to the variety of options that business users can leverage to analyze the data for analytic decision-making. By definition, all data access tools query the data in the data warehouse's presentation area. A data access tool can be as simple as an ad hoc query tool or as complex as sophisticated data mining.

*Metadata:* One of the most important components of data warehouse is metadata. Metadata is all the information in the data warehouse environment that is not the actual data itself. It's actually data about the data. Data warehouse teams often spend an enormous amount of time talking about, worrying about, and feeling guilty about metadata [2]. Since most developers have a natural inclination to the development and orderly filing of documentation, metadata often gets cut from the project plan despite everyone's acknowledgment that it is important. We have operational source system metadata including source schemas and copybooks that facilitate the extraction process. Once data is in the staging area, we encounter source metadata to guide the transformation and loading processes, including staging file and target table layouts, transformation and cleansing rules, conformed dimension and fact definitions, aggregation definitions, and ETL transmission schedules and run-log results. Even the custom programming code we write in the ETL part is metadata. [3]

The data staging area metadata is also being used by the data presentation area and data access tools to control the processes and to do some logic based on the source information. The ultimate goal is to catalog, integrate, and then leverage these disparate varieties of metadata, much like the resources of a library. Suddenly, the effort to build data warehouse models appears to pale in comparison to develop metadata models [2]. However, just because the task is large, one cannot simply ignore the development of a metadata framework for the data warehouse. There is a need to develop an overall metadata plan while prioritizing short-term deliverables, including the purchase or construction of a repository for keeping track of all the metadata.

The purpose of this research is to present a metadata model that can fit in any data warehouse to keep track of source metadata and ETL metadata. This thesis also presents an algorithm for implementation of the ETL job that shows how metadata can be used in data warehousing. The research is trying to help developers by giving them a standard model that they can use and extend to their needs for metadata.

Since more than 70% of data warehousing metadata resides in Source Systems and ETL processes, the research only focuses on these two metadata models [3]. Two preliminary comments are noteworthy. First, Kimball has been active in this area for some time and asserts that general techniques for modeling metadata have not been developed despite a significant level of research activity concerned with data that is either large, distributed or heterogeneous (or all three). In that sense this thesis proposes something that is partly novel. [2]

Certainly it reuses portions of existing expertise on the subject, but the provision of a generalized schema for supporting the use of metadata in unconstrained and unknown contexts is new. Second, it is clear from dialog with other professionals that—perhaps because there is a tendency to overlook the metadata modeling problem—many do not fully understand the role of metadata. Though we will not repeatedly raise this analogy, such reader may look to their knowledge of the type system of a strongly typed language (e.g. ML or Haskell or Ada) to understand the concept and importance of metadata. The type information carried by a type system just is a form of metadata that the compiler uses to ensure robustness and reliability in generated code. We feel that keeping this analogy in mind

can aid in a reader's complete understanding of the material presented here. In the end of chapter four there is a discussion like a use-case scenario that also adds clarity.

## 1.1 MOTIVATION EXAMPLE

Initially, I was motivated to do this research is by thinking about the issues that can occur during the ETL phase of data warehousing. The ETL is the backbone of the data warehouse and consumes more than 70% of the total time [2]. So if we have good ETL system we can think of a successful data warehouse. The following gives our thoughts. Usually, when ETL process runs without metadata structures around it, the following are the typical steps that will be performed:

1. Connect to the source system.
2. Extract the data required into a staging area.
3. Transform the data using hard coded transformations.
4. Load the final table and clean the staging area.

These steps will work fine in a perfect situation, but real situations are never perfect. The following are an issue that has been neglected by building ETL by this process:

1. What if the source system is unavailable? Will the job retry again?
2. What about the mapping information from source to destination? What if the new mapping has to be applied?
3. What if the process fails at some step? Will the process re-run from scratch?
4. What will happen to bad rows? How will the developers know which row is bad that lead to failure?
5. How much time did the process take compared to the time it should take? How is it going to let developers know that the process is not efficiently running?
6. What if during the job schedule, some unplanned outage happens? Would it be possible to change the schedule time for each job running at that time?
7. How will the end user know about all these situations, which might happen?
8. What if one ETL job depends on other, other ETL job should wait for the job it depends on before it kicks off so that it doesn't fail.

So this could become a chain of failures if proper metadata is not defined and loaded when doing ETL.

ETL is a backbone in data warehousing and most of the time data warehousing

projects fail because of bad ETL designs [3]. So in order to solve this issue the research has come up with the metadata model that is applicable to any data warehouse for capturing the metadata at source level and ETL level.

This example was the initial motivation of going deeper into the metadata part of data warehousing and coming up with the model which can help in metadata management.

## **1.2 OBJECTIVE OF THE RESEARCH**

The objective of this research is to realize the importance of metadata in data warehousing. Metadata is always been a challenging topic in the data warehouse world and the research contributes towards creating the standard metadata model. The research tries to solve the following problem:

How much importance needs to be given to metadata modeling during data warehouse development and how can we standardize the metadata model in data warehousing?

## **1.3 RESEARCH METHODOLOGY**

In order to achieve the research objective and try to answer question the following steps has been followed:

1. Study the literature related to data warehouse processes.
2. Study the literature related to metadata for data warehousing and study about the different kinds of metadata that can exists in data warehouse.
3. Develop a platform independent metadata model for source metadata systems and ETL metadata systems.
4. Develop a platform independent algorithm for showing the implementation of the metadata model.
5. Implementation of the model and algorithm in SQL Server database along with the test data to show what can be leveraged out of this model.

## **1.4 OVERVIEW OF THE THESIS**

Chapter 2 discusses the literature based on data warehouse concepts, the details about main components, ETL processes, dimensional modeling and most importantly the metadata structure in data warehousing. Chapter 3 presents the metadata models for source metadata model and ETL metadata model. It also presents the algorithm that shows the way to implement the metadata model. Chapter 4 shows the test results and screen shots associated

with the tests that have been performed. Chapter 5 discusses about the conclusions of the research and possible future work.



## **CHAPTER 2**

### **INTRODUCTION TO DATA WAREHOUSING AND METADATA**

In this chapter, a brief overview of data warehousing and metadata are given. Data warehousing aims at providing integrated decision support reporting by integrating data from various heterogeneous sources and putting them together. The main goal of this chapter is to introduce the data warehouse terms which are used in the rest of the thesis, and to describe the general context of the particular issues treated in this thesis, i.e., the issues related to the data warehouse metadata management. To understand the model that is presented, it is important to understand the engine of the model. For this purpose, the data warehouse architecture, including the main components of a data warehouse system used for data acquisition, data storage and data analysis must be closely analyzed. A special focus of this chapter is the description of the ETL processes, and the concepts used for modeling and storing warehouse data.

#### **2.1 WHAT DATA WAREHOUSE REALLY IS AND ISN'T?**

Data warehousing is the process of taking data from legacy and transaction database systems which are also known as source systems and transforming it into organized information in a user-friendly format and loading into a data store to encourage data analysis and support business decision making. The process that involves transforming data from its original format to a data store is also known as ETL process. Studies have shown that designing and developing the ETL processes takes at least 70 percent of the time, effort, and expense of most data warehouse projects. [2]

According to Ralph Kimball:

A data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making. [3]

In other words, a data warehouse is a home for your company's data, that originates in other source applications, for example: For a company one of the source system for data

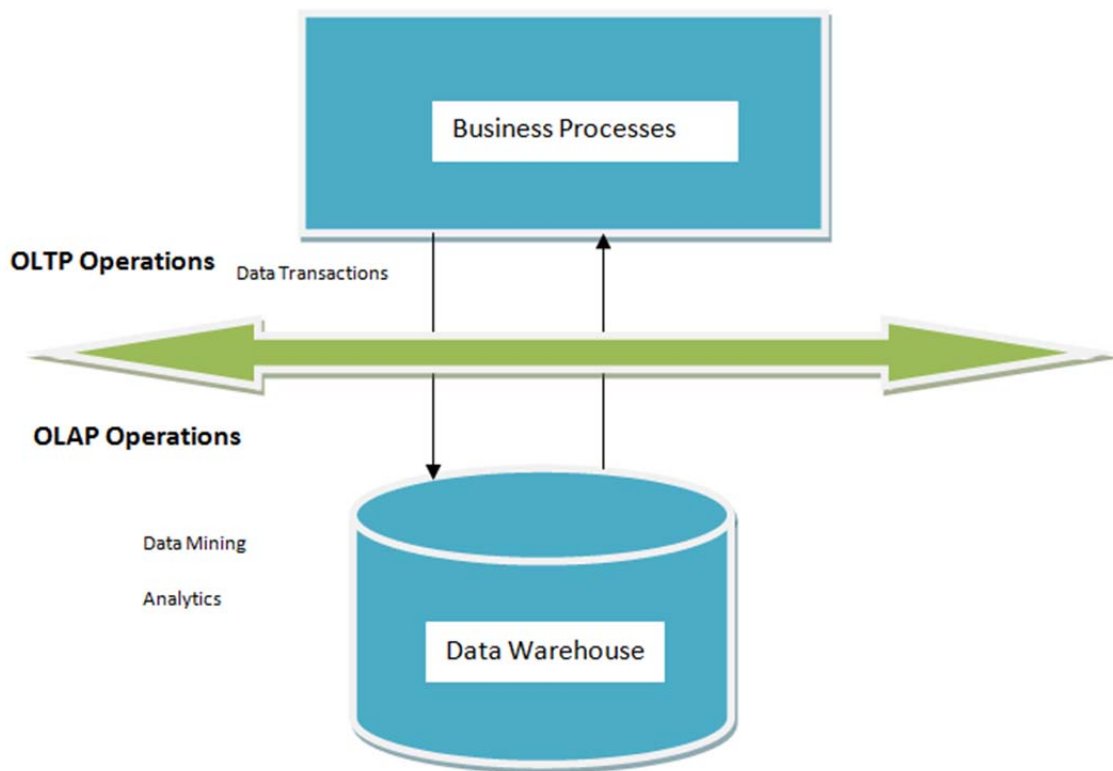
warehouse could become the system which is used to fill customer orders for its products, or some data source external to your company, such as data from credit card payment processing vendors.

Now we have a high level understanding of what data warehousing is all about. But there is another important thing, which we should understand is what data warehousing isn't about which clears many misconceptions that people generally have about data warehouse. To clear up any misconceptions, an individual who is going to be part of a data warehouse team, especially on the ETL side, must know his or her boundaries. The data warehouse environment consists of different processes that together make a data warehouse. The most important thing to remember is that any alone process cannot make a data warehouse. Studies have shown that people struggle with the same misconceptions over and over again. The top five things the data warehouse is mistaken to be are as follows [3]:

1. A product. Contrary to many vendor claims, a data warehouse can never be bought. A data warehouse includes source data analysis, data cleansing, data movement, and finally dimensional modeling and data access. No single product can achieve all of the tasks involved in building a data warehouse. So, calling data warehouse a product is not appropriate. [2]
2. A language. One cannot learn to code a data warehouse in the way programming languages or database languages are learned. The data warehouse is composed of several components, each likely to require one or more programming or data-specification languages. [2]
3. A project. A properly deployed data warehouse consists of different phases of projects. Studies have shown that any attempt to deploy a data warehouse as a single project will almost certainly fail. Each phase of data warehouse is typically considered a separate project with its own timeline [4]. A crucial factor is that each phase contains its own requirements and deliverables that each integrates into a single cohesive unit—the enterprise data warehouse. A better way to think of a data warehouse is as a process, not as a project.
4. A data model. A data model alone does not make a data warehouse. The data warehouse is comprehensive process that, by definition, must include the ETL process. After all, without data, even the best-designed data model is useless. [3]
5. A copy of a transaction system. A common mistake is to believe copying your operational system into a separate reporting system creates a data warehouse. Just as the data model alone does not create a data warehouse, neither does executing the data movement process without restructuring the data store. Both data and data model is required to have a successful data warehouse.

## 2.2 OLTP AND OLAP

IT systems can be divided into transactional (OLTP) and analytical (OLAP). An overview of OLAP and OLTP systems is also shown in Figure 2.1. OLTP systems provide source data to data warehouse, whereas OLAP systems help to analyze it. [5]



**Figure 2.1. An overview of OLTP and OLAP system.**

OLTP (On-line Transaction Processing) is characterized by a large number of short on-line transactions (INSERT, UPDATE, and DELETE). OLTP systems make sure a very fast query processing, maintaining data integrity in multi-access environments and time taken to execute number of transactions. [5]

OLAP (On-line Analytical Processing) is characterized by relatively low volume of transactions. Queries are often complex and involve aggregations. OLAP databases have aggregated and historical data that is stored in star schema or snowflake schema. [5]

The following summarizes the major differences between OLTP and OLAP system design. [5]

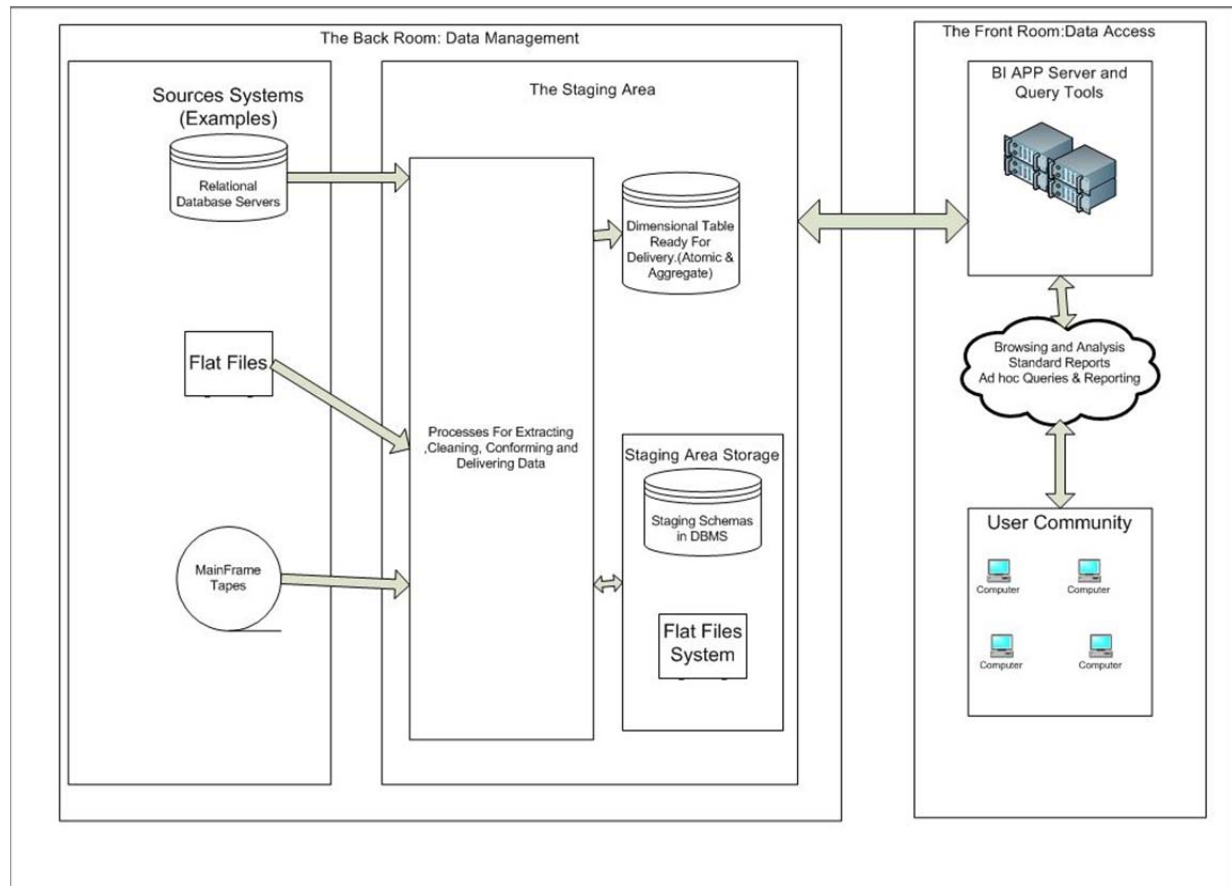
- **Source of Data** - OLAP gathers data from multiple systems (including OLTP systems). OLTP records data. OLAP gets data periodically from backend systems (such as OLTP). OLTP is updated regularly as transactions that run by the application layer.
- **Purpose** – Purpose of OLTP is to carry out application needs. OLAP provides the decision-making queries and results that are analyzed for business needs.
- **Reporting** – OLTP is not a big reporting layer and the amount of data is usually less. OLAP queries typically run on huge volume of data.
- **Resource requirements** – OLAP requires huge space and CPU resources to store volume of data and run complex queries. OLTP requires relatively less system resources.
- **Execution Speed** – OLTP runs faster than OLAP as queries tends to be simple.

## **2.3 ARCHITECTURE OF DATA WAREHOUSE**

The main goal of this section is to introduce the standard architecture of data warehouse. There are three main layers in any typical data warehouse. The first layer is called as the source systems layer that is a source for data warehouse, the second layer is called as staging area where the data from source system is extracted, transformed and loaded into the data store and the third layer is called as reporting layer that is a front end for the user to see the data [6]. These three layers are can also be divided into two layers called the back room and the front room. Back room contains the source system, ETL processes and data store, front room is the way the data from the data store is accessed (OLAP Cube, Reporting, Dashboards). Figure 2.2 shows the two distinct components of a typical data warehouse. [3]

### **2.3.1 The Back Room**

The back room is also known as data management because all the data management related processes are done in this area. The back room and the front room of the data warehouse are physically, logically, and administratively separate [6]. In other words, in most cases the back room and front room are on different machines, depend on different data structures, and are managed by different groups. The major operations that are performed in the back room involve acquiring data and transforming it into information, ultimately delivering that information to the query-friendly front room. We assume that data access is



**Figure 2.2. A typical data warehouse architecture.**

prohibited in the back room, and therefore the front room is dedicated to just this one purpose. [7]

This architecture is a great way of separating the task of the complex logic of ETL from the reporting layer. This way the dependency of logic layer and presentation layer is decoupled. Ralph Kimball has explained a very good example on this approach:

Think of a restaurant. Imagine that patrons of the restaurant are end users and the food is data. When food is offered to patrons in the dining room, it is served and situated exactly as they expect: clean, organized, and presented in a way that each piece can be easily identified and consumed. [3]

Meanwhile, before the food enters the dining room, it is prepared in the kitchen under the supervision of an experienced chef. In the kitchen the food is selected, cleaned, sliced, cooked, and prepared for presentation. The kitchen is a working area, off limits to the patrons of the restaurant. In the best restaurants, the kitchen is completely hidden from its customers- exposure to the kitchen, where their food is still a work-in-progress, spoils the customer's

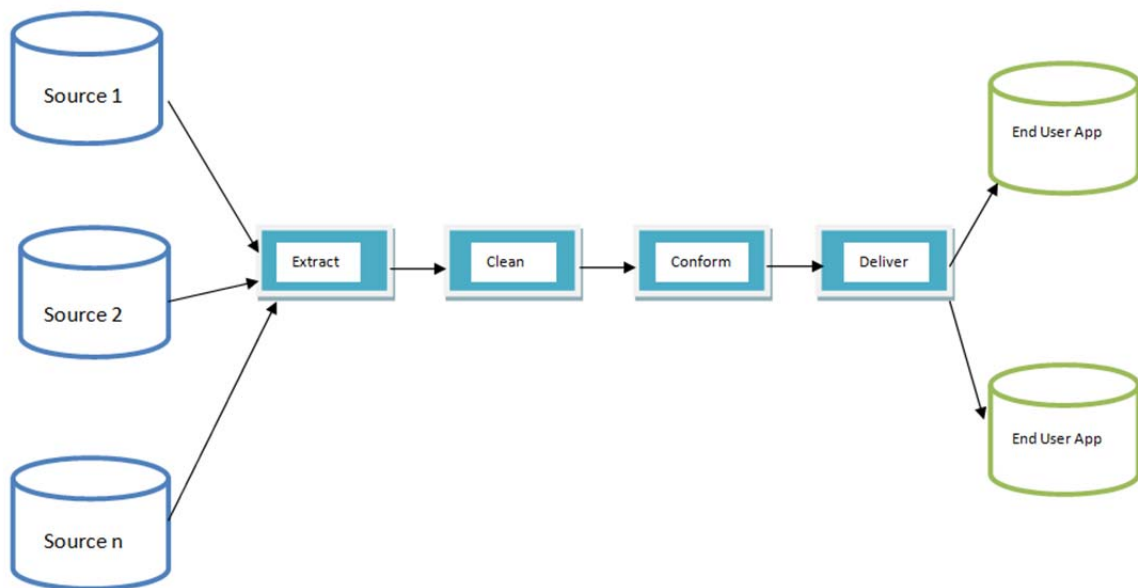
ultimate dining experience. If customer requests information about the preparation of food, the chef must come out from the kitchen to meet the customer in the dining room-a safe, clean environment where the customer is comfortable-to explain the food preparation process. [3]

The staging area acts as a kitchen of the data warehouse. It is a place accessible only to development team. It is a back-room facility, completely off limits to end users, where the data is placed after it is extracted from the source systems, cleansed, manipulated, and prepared to be loaded to the presentation layer of the data warehouse. Any metadata generated by the ETL process that is useful to end users must come out of the back room and be offered in the presentation area of the data warehouse. [8]

Prohibiting data access in the back room kitchen relieves the ETL team from:

- Providing detailed security at a row, column, or applications level
- Building query performance-enhancing indexes and aggregations
- Providing continuous up-time under service-level agreements
- Guaranteeing that all data sets are consistent with each other

There are four major steps found in almost every data warehouse, as shown in Figure 2.3, every back room system supporting the data warehouse is structured with these four steps [8]. The four steps are (see Figure 2.3):



**Figure 2.3. The four back room steps of a data warehouse (ETL).**

*Extracting:* The data extracted from the source systems is usually written directly to disk before significant content transformation takes place. Data from source systems often is written to flat files or relational tables in this step. This allows the original extract to be as simple and as fast as possible and allows greater flexibility to restart the extract if there is an interruption. In some cases, initially captured data is discarded after the cleaning step is completed, and in other cases data is kept as a long-term archival policy. [3]

*Cleaning:* Cleaning the data is important to transfer the data from staging to the multi dimensional schema. Data cleaning may involve many steps, including checking for valid data, ensuring consistency across data, removing duplicates, and checking whether complex business rules and procedures have been enforced. The results of the data-cleaning step are often saved semi permanently because the transformations required are important and standard. [3]

*Conforming:* Data conformation is required whenever two or more data sources are merged in the data warehouse. Separate data sources cannot be queried together unless some or all of the sources have been made identical and unless similar data have been mathematically rationalized so that differences and ratios between the data make sense. Data conformation is a significant step that is more than simple data cleaning. [3]

*Delivering:* The aim of back room is to make the data ready for presentation. This back room step is structuring the data into a set of simple, symmetric schemas known as dimensional models known as star schema or snowflake schema. These schemas significantly reduce query times and simplify application development. Dimensional schemas are a necessary basis for constructing OLAP cubes or any other end user application like Microsoft reporting services, dashboards. [3]

### **2.3.2 The Front Room: Data Access**

Accessing data in the front room of the data warehouse is a project that must be closely coordinated with the building and managing of the ETL system. The different ways in which data can be accessed are by writing query, reporting tools, dashboards, and OLAP cubes [9]. The data in the front room is what end users see. Data marts are an important component of the front room. Data marts are nothing but measures which are important for business which is surrounded by descriptive entities like Product, Customer, Geography etc,

for example Number of sales occurring in a particular region of a country can be accessed through a data mart. [10]

### **2.3.3 OLAP**

OLAP (On line Analytical Processing) is a way of denormalization of data into a data model that can be used for business analysis [11]. In comparison to OLTP systems, OLAP is highly interactive and dynamic. The main usage of OLAP is by creating an OLAP Cube and accessing data from it. An OLAP cube or simply cube is a multidimensional data structure from which you can query for business information residing on analysis server or any data store. A cube can contain fact data from one or more fact table. Fact table contains measures that are the numbers that are analyzed. They are found in the fact table; examples include total sales and total profit. A fact table is also known as a measure group. OLAP cube also has something called dimensions that are the business entities that qualify what you are analyzing. For example analyzing total sales BY date, Profit BY State. The term after BY is nothing but dimensions that is typically by which business can slice and dice the data. [9]

Figure 2.4 shows an example of a multidimensional cube. Each block of data is called as cell. A cell is an entity from which you can retrieve data that is pertinent to an intersection of all the dimension members [12]. Cells represent the numeric measures which is also known as facts corresponding to the objects of analysis, e.g., sales, orders, revenue and all these measures can be sliced by Quarter (Time dimension), Product line and Geography (Country).

Apart from OLAP cube there are other tools and technologies that can be used for data access. Data can also be accessed to create Crystal Reports, Business Intelligence Dashboards. [12]

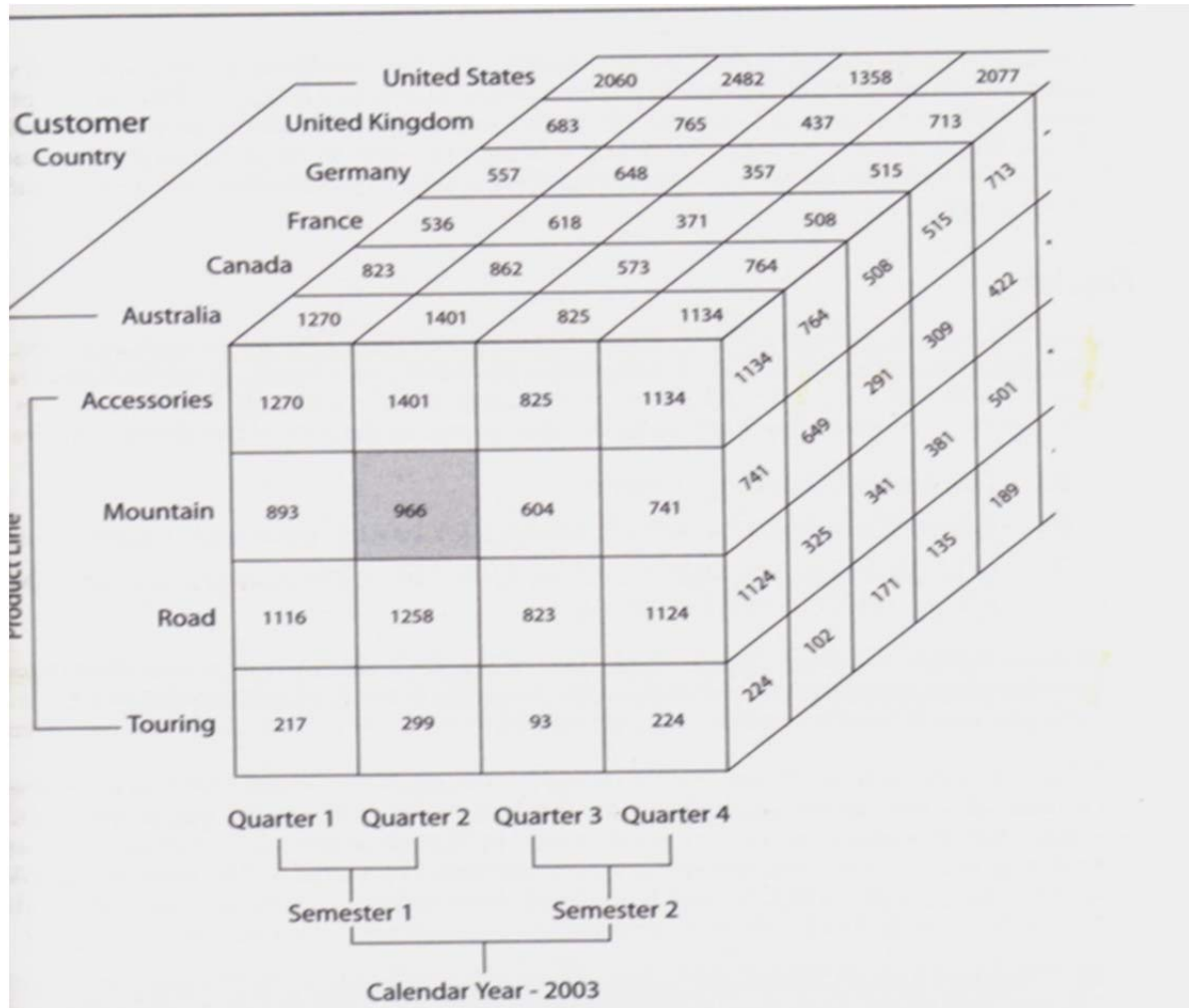
## **2.4 DATA WAREHOUSE MODELING**

There are four major steps that data modeling has to go through during its lifecycle. We will be discussing them briefly in following sections.

### **2.4.1 Conceptual Data Model**

A conceptual data model identifies the highest-level relationships between the different entities. Features of conceptual data model include [13]:





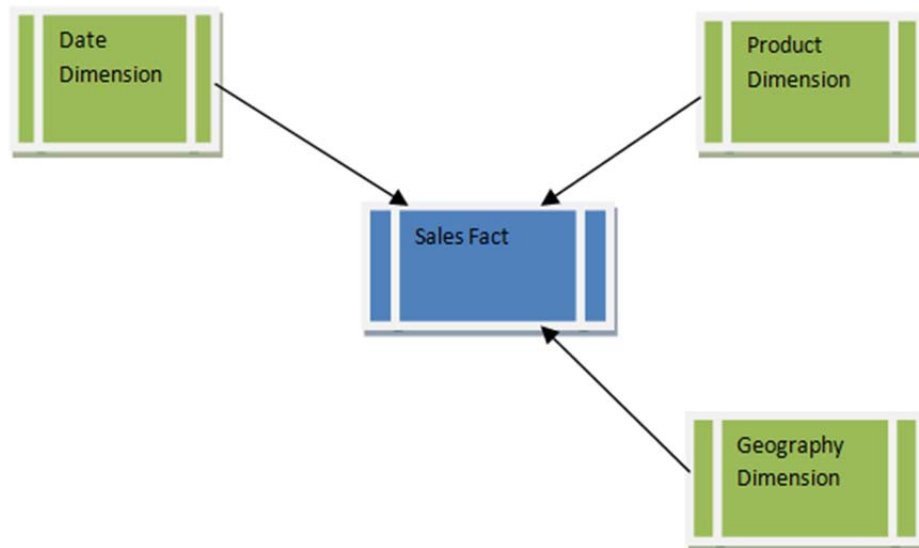
**Figure 2.4. An example of multidimensional data cube.**

- Includes the important entities and the relationships among them.
- No attribute is specified.
- No primary key is specified.

From Figure 2.5, we can see that the only information shown via the conceptual data model is the entities that describe the data and the relationships between those entities. No other information is shown through the conceptual data model.

### 2.4.2 Logical Data Model

A logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Features of a logical data model include [13]:



**Figure 2.5. Example of conceptual data model.**

- Includes all entities and relationships among them.
- All attributes for each entity are specified.
- The primary key for each entity is specified.
- Referential Integrity is specified (FK Relation).

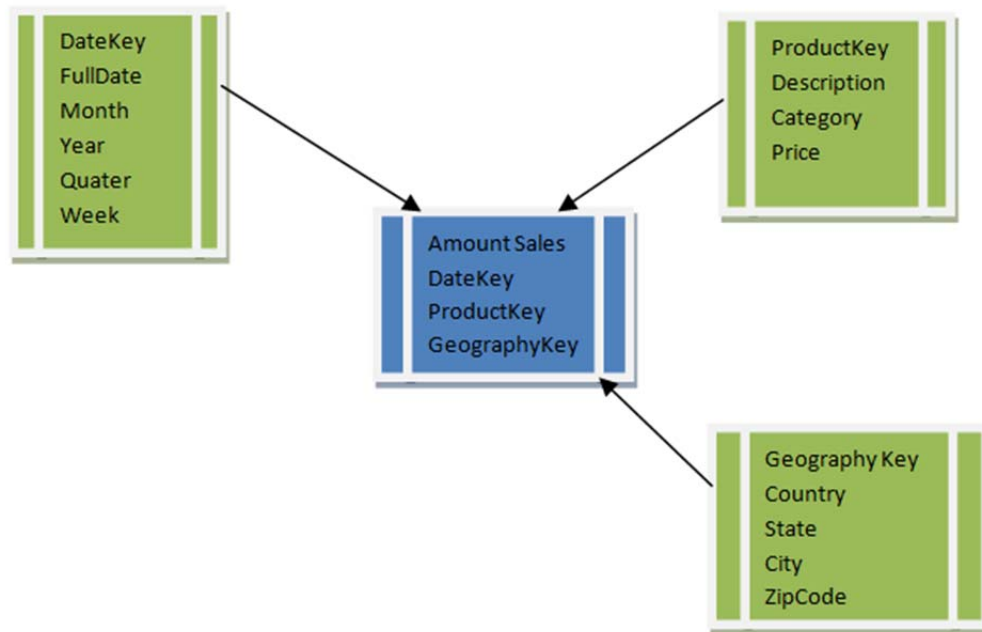
The steps for designing the logical data model are as follows [13]:

- Specify primary keys for all entities.
- List the relationships between different entities.
- List all attributes for each entity.
- Normalization.
- No data types are listed

The Figure 2.6 is an example of a logical data model.

### 2.4.3 Physical Data Model

Physical data model represents how the model will be presented in the database. A physical database model shows all table structures, column names, data types, constraints, primary key, foreign key, and relationships between tables. Features of a physical data model include [13]:



**Figure 2.6. Example of logical data model.**

- Specification all tables and columns.
- Foreign keys are used to identify relationships between tables.

The steps for physical data model design are as follows [13]:

- Convert entities to tables.
- Convert relationships to foreign keys.
- Convert attributes to columns.

Figure 2.7 is an example of a physical data model.

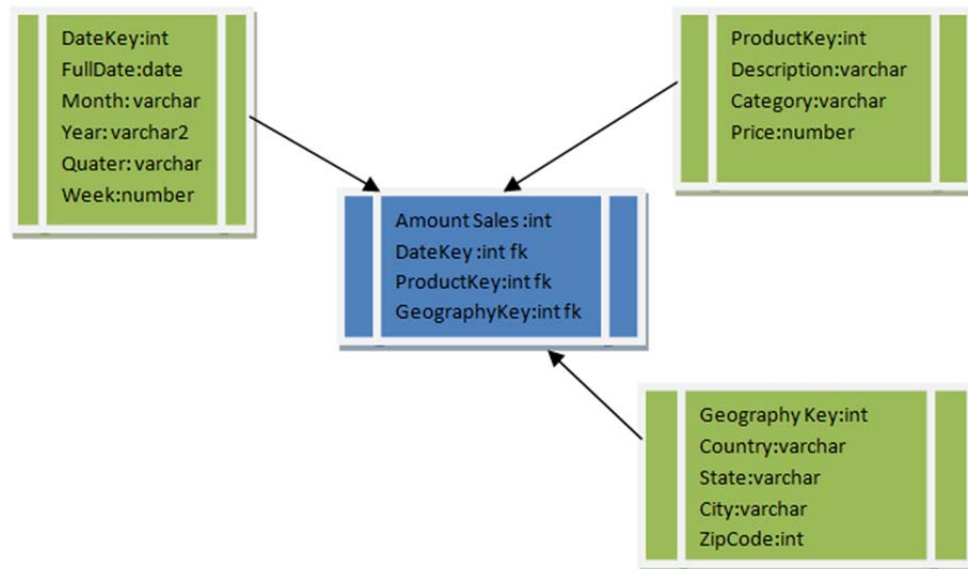
#### 2.4.4 Dimensional Data Model

Dimensional data model is most often used in data warehousing systems. This is also known as OLAP system.

To understand dimensional data modeling, the following terms need to be understood [13]:

*Dimension:* A category of information. For example, the time dimension.

*Attribute:* A grain within a dimension. For example, month is an attribute in the time dimension.



**Figure 2.7. Example of physical data model.**

*Hierarchy:* The specification of levels that represents relationship between different attributes within a dimension. For example, one possible hierarchy in the time dimension is Year, Month, Quarter, and Date.

*Fact Table:* A fact table contains the measures. For example total number of sales, total profit. The measure is stored in the fact table with the appropriate granularity. For example, it can be sales amount by store by week. In this case, the fact table would contain three columns: A date column, a store column, and a sales amount column.

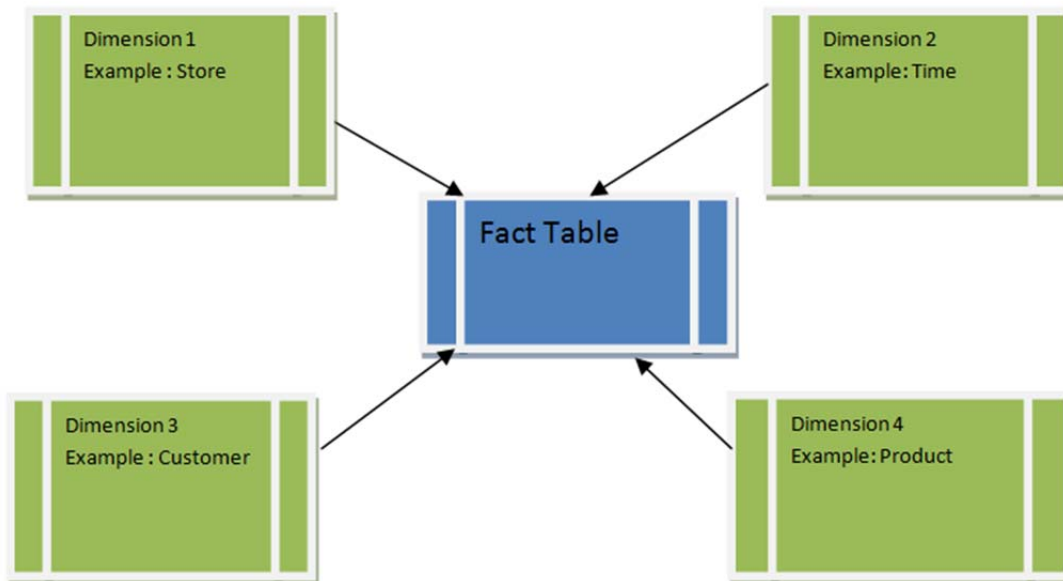
*Dimensional Table:* The dimensional table provides all the detail level information about the attributes. For example, the dimensional table for the quarter attribute would include a list of all of the quarters available in the data warehouse. For example, first quarter of 2012 may be represented as "Q1 2012" or "2012 Q1".

A dimensional model includes fact tables and dimensional tables. A fact table is connected to dimensional tables via referential integrity, but fact tables do not have direct relationships to one another. In designing data models for data warehouses, the most commonly used schema types are **Star Schema** and **Snowflake Schema**.

In the star schema design, the fact table sits in the middle and is connected to other dimension lookup tables like a star. Each dimension is represented as a single table. The

primary key in each dimension table is related to a foreign key in the fact table. Figure 2.8 shows an example of star schema.

Example: Assume our data warehouse keeps store sales data, and the different dimensions are time, store, product, and customer. In this case, the figure on the left represents our star schema. The lines between two tables indicate that there is a primary key / foreign key relationship between the two tables. Note that different dimensions are not related to one another (see Figure 2.8).

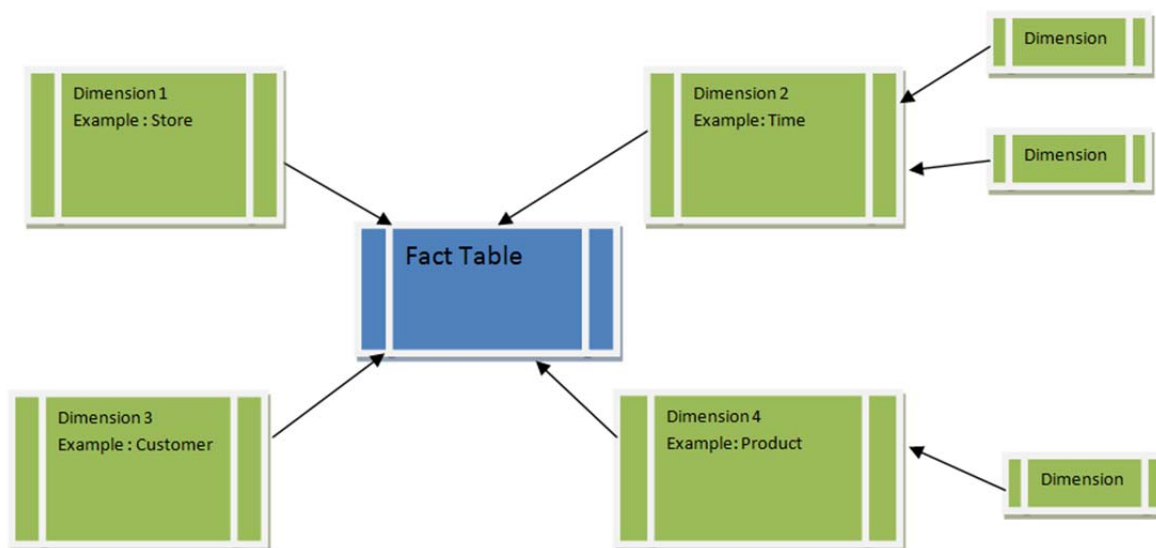


**Figure 2.8. An example of star schema.**

The snowflake schema is an extension of the star schema (see Figure 2.9), where each point of the star explodes into more points. In a star schema, each dimension is represented by a single dimensional table, whereas in a snowflake schema, that dimensional table is normalized into multiple lookup tables, each representing a level in the dimensional hierarchy. [13]

## 2.5 DATA WAREHOUSE METADATA

Most people think that data warehouse is a large collection of historical, integrated data. While that thinking is correct in many ways, there is another very important element of the data warehouse that is vital- metadata. The thesis main focus is on metadata importance



**Figure 2.9. Example of snowflake schema.**

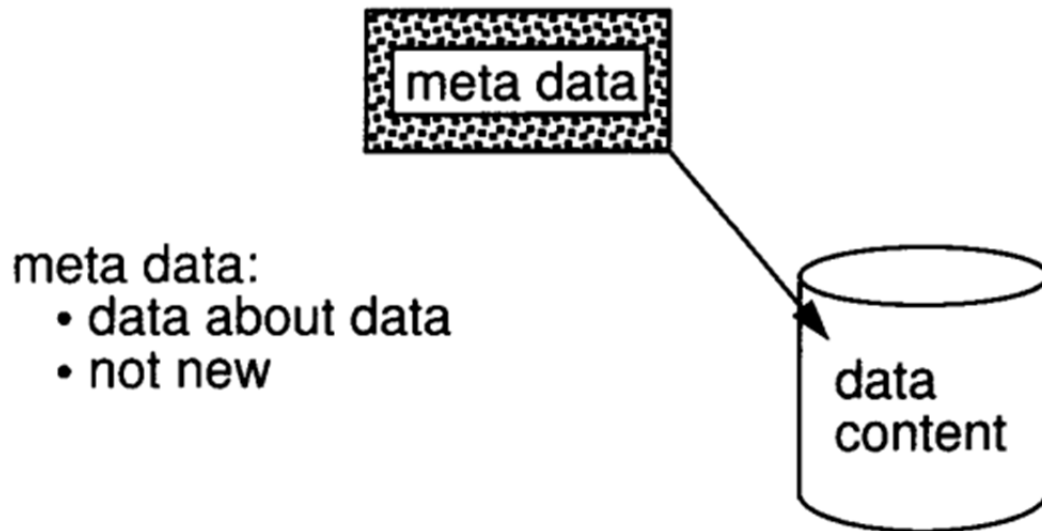
in data warehousing. Metadata means data about the data, that can be very hard to understand at first. This thesis will help in bringing out the real importance and relationships of metadata with data warehousing in further chapters. This section is dedicated towards brief overview of metadata in data warehouse.

Metadata is also an interesting topic because every tool in the data warehouse world including business intelligence (BI) tools, ETL tools, databases, and dedicated repositories claims to have a metadata solution. According to Ralph Kimball *"Even after years of implementing and reviewing data warehouses, we've yet to encounter a true end-to-end metadata solution"* [3]. The main focus of the thesis is to propose a set of metadata structures that supports the data warehouse.

According to Inmon Metadata Paper:

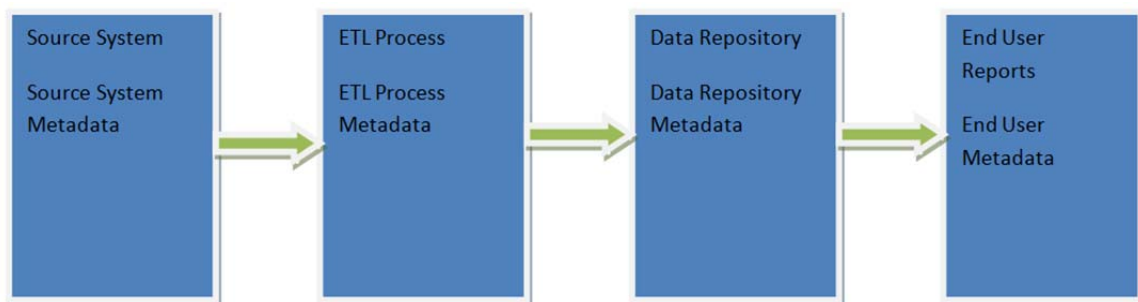
While metadata is not new, the role of metadata and its importance in the face of the data warehouse certainly is new. For years the information technology professional has worked in the same environment as metadata, but in many ways has paid little attention to metadata. The information professional has spent a life dedicated to process and functional analysis, user requirements, maintenance, architectures, and the like. The role of metadata has been passive at best in this milieu. [6]

Figure 2.10 [8] shows the metadata in its most simple form. There are three levels in data warehouse where metadata should be captured. The first is the source system metadata, second is the back room metadata which involves metadata related to all the mapping



**Figure 2.10. Inmon metadata model.** Source: R. KIMBALL, *A dimensional modeling manifesto*, J. Data Base Mgmt. Syst., 10 (1997), pp. 58-70.

between source and destination or ETL processes relations, and third is the metadata related to the front room also called as business metadata. The thesis will present a standard way of capturing metadata in the back room that can easily be extended for front room (see Figure 2.11).



**Figure 2.11. Data warehouse metadata model approach.**

According to Ralph Kimball:

Back room metadata presumably helps the DBA bring the data into the warehouse and is probably also of interest to business users when they ask where the data comes from. Front room metadata is mostly for the benefit of end user, and its definition has been expanded to not only be the oil that makes our tools function smoothly but a kind of dictionary of business content represented by all data elements. [3]

### 2.5.1 Source System Metadata

Source system metadata is one that belongs to source. The sources could be mainframes, flat files, third party data providers, or even on-line sources. All we do here is read source data and extract it to a data staging area. The following are the different types of metadata that can be captured from source systems. [3]

Source Information:

- Repositories
- Source schemas
- Data Format
- Relational source system tables and DDL

Source Description Information:

- Ownership descriptions of each source
- Business descriptions of each source
- Update frequencies of original sources
- Access rights privileges, roles

Process Information:

- Job schedules
- Extraction of data
- ETL information

### 2.5.2 Back Room Metadata

Now let's list all the metadata needed to get data into a data-staging area and prepare it for loading into dimensional model. In back room of the data warehouse there are different processes that runs, the data has to go through extraction, transformation process, and then the data needs to be loaded into the dimensional data model .So during this process the metadata is attached to all the individual processes and also includes the metadata. Following is the list of all different set of metadata that needs to be captured:

Data Model Metadata:

- Definitions of facts and dimensions
- Job details for sources and look up attributes
- Slowly changing dimension policies for each new attribute (for example overwrite create new record or create new field)



ETL Metadata:

- Data cleaning rules
- Mapping information
- Required Transformations
- Target schema designs, source to target data flows

Monitoring ETL Metadata:

- Audit records
- ETL job run time logs, summaries, and time stamps
- Business descriptions of extract processing
- Data-staging area archive logs and recovery procedures
- Stored procedures and SQL administrative scripts
- DBMS backup, status-backup procedures and backup security.

### **2.5.3 Front Room Metadata**

The front room metadata is one that is associated with users of data warehouse. Some of the metadata associated with it are [1]:

- Business names and descriptions for columns tables groupings
- Pre scanned query and report definitions
- End user documentation and training aids
- Network security user privilege profiles
- Network security authentication certificates
- Network security usage statistics, including log on attempts access attempts and user ID by location reports
- Usage and access maps for data elements, tables, and views reports

Now it is cleared that what metadata is all about and why it is so important. It is everything, except for the data itself. In a sense, metadata is the DNA of the data warehouse. It defines all elements and how they work together and with the help of well-defined metadata we can control the whole data warehouse processes. The main aim of our thesis is also to present a methodology by which we can capture and control data warehouse.

## CHAPTER 3

### METADATA METHODOLOGY OF DATA WAREHOUSING

#### 3.1 INTRODUCTION

This chapter will present a metadata model for data warehousing. The discussions in chapter 2 have lead to a good understanding of what metadata is and how important it is to develop and design it in any data warehouse. The methodology revolves around the idea that during all the phases of data warehouse, metadata is an integral part and is always coupled to the data. The model presents metadata as a shield to all the data warehouse phases. This chapter first presents metadata methodology for data warehousing, and in further sections takes a deep dive in presenting the platform independent metadata model architecture.

#### 3.2 METADATA MODEL FOR DATA WAREHOUSING

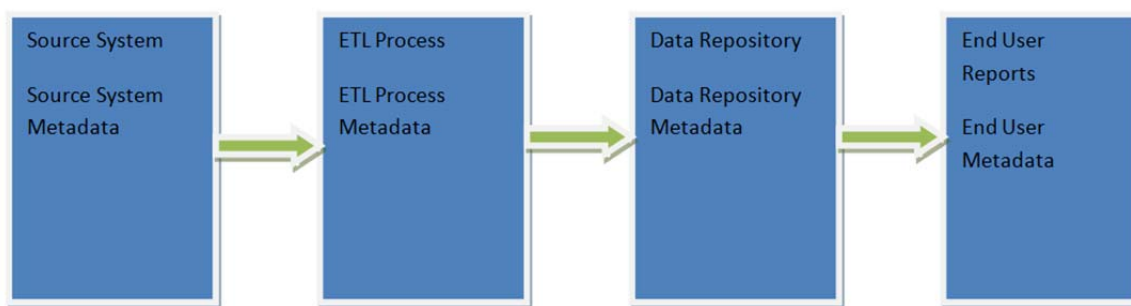
As previously discussed in chapter 2, there are three main parts of data warehouse:

- *Source System*: The source system acts as a source for the data in data warehouse. Typically, there are multiple source systems from which data needs to be either pulled or pushed to data warehouse. Examples of source systems are legacy systems, flat file data source, DB2 source system. The data from source systems is extracted and loaded into back room of data warehouse where it is cleaned and loaded into dimensional. The main user of source systems is an ETL team.

The thesis main focus is on metadata, so ETL team needs to understand the metadata about the source systems to build the ETL process. The model suggests that a layer of metadata has to exist around the source systems. The details about the metadata will be discussed in later sections. Figure 3.1 shows a data warehouse metadata model.

- *ETL Stage (Back Room)*: The primary focus of the ETL stage is to extract the data from source systems, then transform the data so that it can become compatible with the data warehouse system and then finally load the data into data warehouse model.

The metadata involved with ETL stage is of various type and is dependent on source system metadata. The source system metadata feeds the metadata for ETL processes. There is a dependency on ETL metadata by source system metadata [14]. An example of source system metadata when source system X is not available then the ETL process metadata



**Figure 3.1 Data warehouse metadata model (high level thesis approach).**

would know that source X is unavailable, so the ETL jobs which depend on source X will wait until the source is available or some business logic can be applied. This example shows how much metadata can help in making ETL smarter and handle exceptions. So the model suggests a ETL system metadata layer. (Figure 3.1)

- *Front Room Systems (Reports, Dashboards, and BI):* The front room is the place where the business end user gets the access to the data. The data can be accessed by creating Reports, Dashboards, BI Scorecards. [14]

The data access is done through automated tools. Examples of some tools are Cognos, SQL server Analysis services, OBIEEEE. These tools take care of most metadata requirements but it is advisable to store that metadata in database for analysis. An example of why we would need this is explained here: Suppose an end user runs a report, he/she should know where the data is coming from and whether the data is up to date or not. There could be a situation when data warehouse is down, so the user should know about it. These issues can be dealt very efficiently if there is a good metadata model around front room layer of data warehouse.

So the model suggests a front room metadata layer (Figure 3.1).

### 3.3 METADATA MODEL DETAILS

There are two major metadata models that will be presented and these are the heart of the thesis discussion. Both models are platform independent and can be extended.

#### 3.3.1 Source Metadata Model

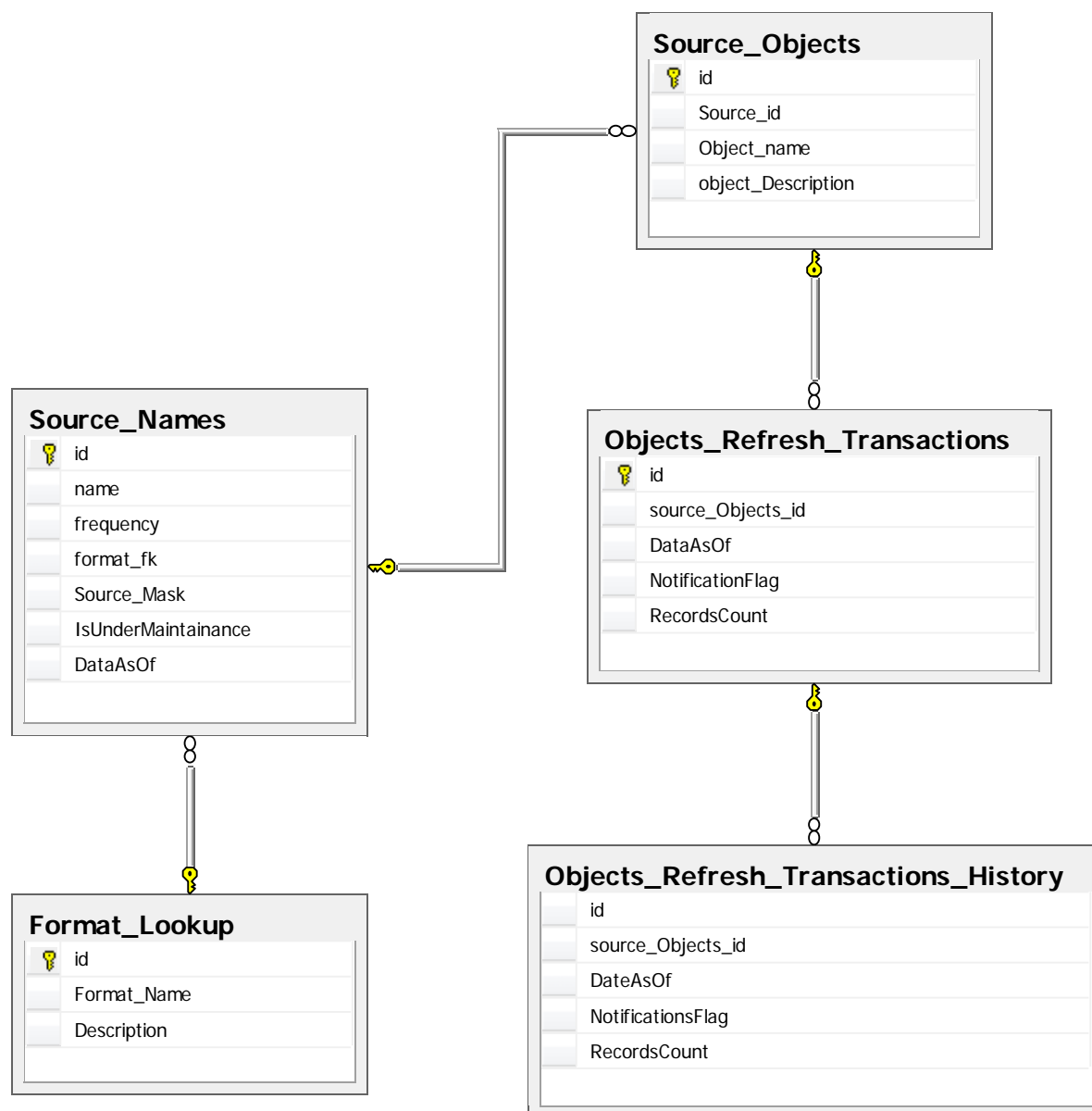
The ETL team needs to know every intimate detail of each table it accesses for the data warehouse. Suppose that one needs to hire for an open position on the team. Before one

can hire, an HR department has to find candidates, prescreen them, introduce them, and arrange for formal interviews. During the interviews, one can identify any weaknesses that might need some work before candidate is chosen for the position. When data warehouse of loaded, the data modeler creates the position, the data warehouse architect finds and prescreens the source. Then data must be analyzed thoroughly to identify its weaknesses, and a plan of action must be initiated to make data satisfactory enough to be included in the data warehouse. Some data will be perfect, other data elements will need some transformation to make sure the quality of data is maintained. [2]

When source systems are analyzed, metadata is required. Figure 3.2 shows the source metadata model. This model can be applied to any number of sources and is generic in nature.

Following is the explanation of the source metadata model:

1. *Source\_Names*: This table in the model contains the information about the sources of the data for data warehouse. The attributes are:
  - *id*: This is the primary key for the this table
  - *name*: It is the name of the source system from which the data is extracted.  
Example: Source could be a database, flat file, Microsoft Excel.
  - *Frequency*: This attribute tells us about the frequency at which the data will be available. Example: A flat file is available from source once a day.
  - *format\_fk*: This foreign key will come from the lookup table *Format\_Lookup* and is the format of the data. The lookup table is always better because we can have different formats and avoids unwanted values in this field.
  - *Source\_Mask*: The *source\_mask* gives information about the mask of the source that can help in automating the ETL job. For example if many flat files are coming from different sources then using this attribute one can figure out the source information.
  - *IsUnderMaintainance*: This flag will tell us that if there is any maintenance going on in the source system. This information is very helpful because ETL job would know that data is not present due to some issue on the source side, so logic can be placed to wait for the job until the source is available.
  - *DataAsOf*: The *DataAsOf* attribute provide information about the last refreshed time for the data from source system. So if the data in the source system needs to



**Figure 3.2. Source metadata model.**

be refreshed every hour then the latency between the current time and this attribute cannot be greater than one hour. If it is greater than one hour, then we know that there is some issue on source side.

2. *Source\_Objects*: This table provides more information about the source. Every source can have multiple objects from which data can be pulled. An example is, from one database called point of sale, four tables and two views is acting as a source for our ETL process. This information for all those objects is present in this table that has the fk relation with the source it is extracted from.
3. *Object\_Refresh\_Transactions*: This table provides the information about availability of the data on the source system to be used by ETL system. This is a transaction level information for all the sources objects and provide necessary information needed to

start an ETL job. The DataAsOf field and notification flag provide information to the ETL job to be executed. The record count acts as control information to check if the row from source matches what has been loaded. This could help in validating the load count. So this table has the latest row for each source object. Anytime a new row is inserted, the previous row will be moved to the history table.

4. *Object\_Refresh\_Transactions\_History*: This entity as explained in the '3)' is the history of all the objects transactions that have occurred.

### 3.3.2 ETL Job Metadata Model

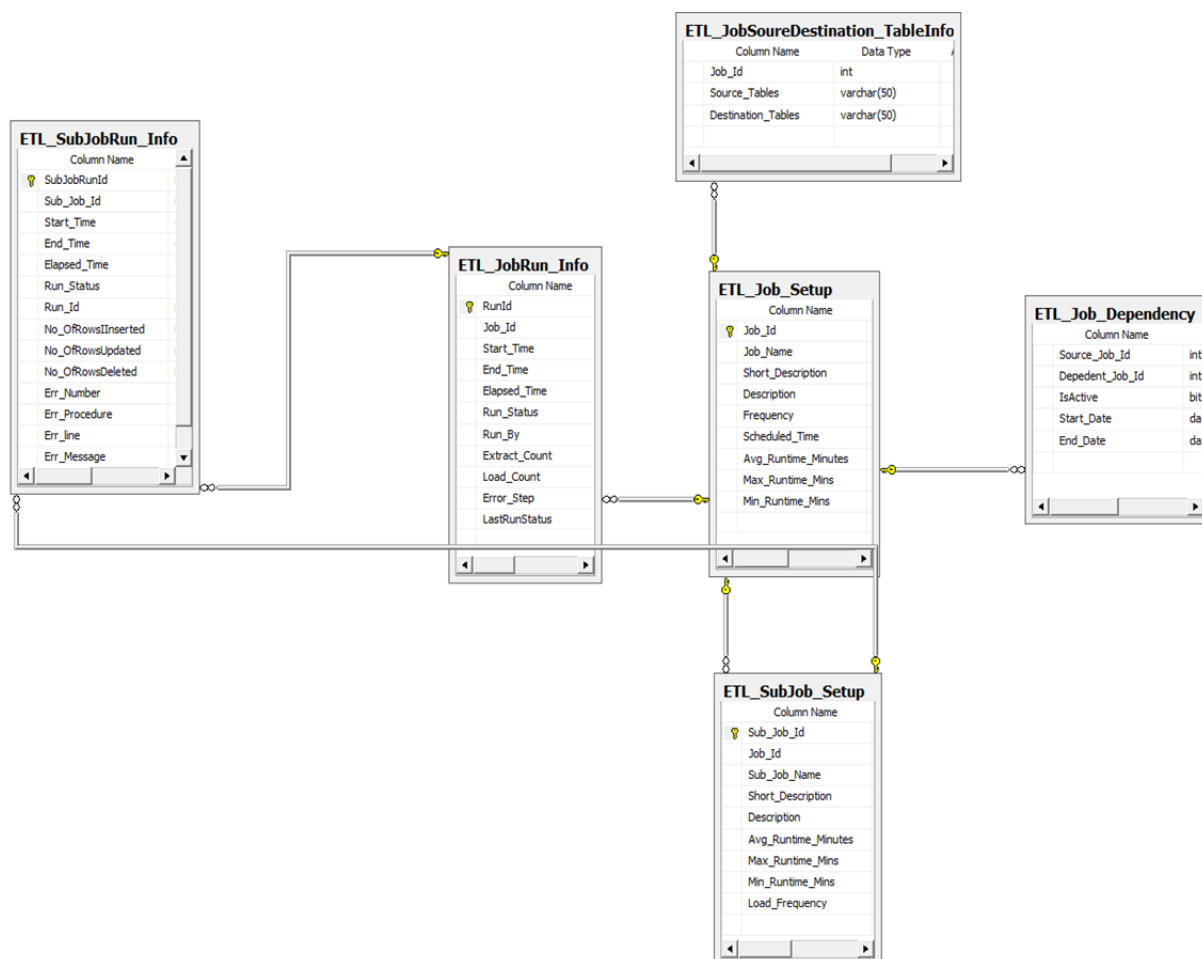
The previous section explained metadata created outside of the ETL system. This section addresses metadata generated by the ETL system and used either within the ETL to manage the ETL processes or by end users or other data warehouse process. The model presented mainly focus on using the metadata from source system and uses it to control the ETL process and captures the metadata for the ETL process that would help in analyzing the ETL part of data warehouse. Specific metadata must be generated to capture the inner workings of each process during ETL. The ETL job is a container that stores all of the transformations that actually manipulate the data. The job metadata is very valuable because it contains the data lineage of the elements in the data warehouse. Every ETL task from extraction to load and all transformations in between are captured in the ETL job metadata. So, all the processes that run under a particular job will also be captured under the ETL job metadata.

The following diagram (Figure 3.3) represents the model for ETL job metadata.

Figure 3.3 shows the job metadata model. This data model is capable of capturing the events that would occur to the job and also acts as a control structure for ETL jobs and define a relationship between them. The following is the detail level explanation of the tables involved in this model:

#### 3.3.2.1 ETL\_JOB\_SETUP

This is center table in this model. The purpose of this table is to have the information about every ETL job that exists in data warehouse. The information about the job contains job\_id which is the primary key for this table, name of the ETL job, description about the job, frequency of the job, scheduled time of the job to run, average time job takes to run, max and min time the job should take to run. This information is very useful when setting up any



**Figure 3.3. ETL job metadata model.**

new job, for example max run time of the job can be used to compare if the job currently takes more time than it should, if that is the case one can have email generated to the DBA to look if they can work on the optimization of the job. This example is useful to understand what can be done if this metadata is available

### 3.3.2.2 ETL\_SUBJOB\_SETUP

The subjob\_setup table contains the information about the steps of the parent job. The ETL job consists of different steps and each different step is called a sub job. So, this table provides sub job information and is a onetime setup whenever a new job is created. This table will have a foreign key relationship with ETL\_job\_setup table on the job\_id column. The other columns for this table are job\_name, job\_description, and frequency of this sub job, time at which it is supposed to run, max and min time it should take to run and average time it should take to run. The ETL job can be controlled on the sub job level by using this

approach that adds more flexibility to extend the process. For Example: if ETL job has 5 sub jobs, and suppose we have the failure at sub job 4, then instead of re-running the job from the step 1 one can design ETL process in such a way that it skips the sub jobs which was done successfully and re-run only the Step 4.

### 3.3.2.3 ETL\_JOB RUN\_INFO

This table will be populated when the ETL job runs. The framework for the ETL job is presented in the next section. The main idea of this table is to see what ETL jobs are running at this moment and information about the most current run for the job. The job\_id in this table is the foreign key to the job\_id in the ETL\_Job\_Setup table. The following are the columns this table contains:

- Run\_Id: This is a primary key for this table and is an identity column.
- Job\_Id: Job\_Id is there so that we can get the information about the job for which this row belongs. This is a foreign key relationship.
- Start\_time: This is a date time field and it is populated at the time when the job run starts.
- End\_Time: This is a date time field and is populated at the time when the job run completes.
- Elapsed\_Time: This is a difference of the Start\_time and End\_time and tells the amount of time that this job took to run. Elapsed\_Time is in seconds.
- Run\_Status: This field will be populated when the job initially starts as 'RUNNING' and depending upon the outcome of the job the status will be changed to 'SUCCESS' OR 'FAIL'. This status tells the most current status of the job.
- Run\_By: This field gives the information about the account under which the job is run. ETL job can be run by either a service account or any other account depending on the privileges. This is very helpful information when we want to track that ran the job.
- Extract\_Count: This field will be populated with the total amount of record count associated with the job run .For example: If the ETL job loads a data from a file, it is a good practice to know how many total records was extracted from file. This info is helpful to compare it with any source metadata to make sure the load is successful, if the metadata for file shows 10,000 records in the file and the load extracts is only 9,000 records, it means there is some issue with the file.
- Load\_Count : Load count gives the count of records which ultimately got loaded. During the load, there could be some bad records from the source. This kind of situation occurs all the time in data warehousing. The difference between the extract count and load count will be the bad record count.



- **Error\_Step:** This field provides the information about the step at which the error has occurred. For example if the ETL has 4 steps, then the step at which the job has failed, this field will be populated. If the job fails at step 3, then the Error\_Step will be 3 and it maps to the Sub JobId.
- **LastRunStatus:** This field provides the last run status of this job. This field is helpful information for an ETL job because it can help control other ETL job depends upon the last run status.

### **3.3.2.4 ETL\_SUBJOBRUN\_INFO**

This table will be populated when the ETL Sub job runs which is a sub step of ETL job and is a part of the ETL job. This table will have more detail level information about each sub job. This table captures all the information about sub job step with the run\_id. This helps figuring out the status of the run for that SubJobId. The following discusses the columns for this table:

- **SubJobRunId:** This is a primary key for this table and will be an identity column.
- **Sub\_Job\_Id:** Sub\_Job\_Id is there so that we can get the information about the job for which this row belongs. This is a foreign key relationship.
- **Start\_Time:** : This is a date time field and it is populated at the time when this sub job run starts.
- **End\_Time:** This is a date time field and is populated at the time when this sub job run completes.
- **Elapsed\_Time:** This is a difference of the Start\_time and End\_time and tells the amount of time that this job took to run. Elapsed\_Time is in seconds.
- **Run\_Status:** This field will be populated when the job initially starts as 'RUNNING' and depending upon the outcome of the job the status will be changed to 'SUCCESS' OR 'FAIL'. This status tells the most current status of the job.
- **Run\_Id:** Run\_Id is a foreign key to the Run\_Id in the ETL\_Jobrun\_Info table. This run\_id provide the information about the ETL job running of which this Sub Job is one of the step.
- **No\_OfRowsInserted:** This field is populated when the Sub job inserts the rows and this will tell the count of the rows inserted in this job.
- **No\_OfRowsUpdated:** This field is populated when the sub job updates the rows and this will tell the count of the rows updated in this job.
- **No\_OfRowsDeleted:** This field is populated when the Sub job deletes the rows and this will tell the count of the rows deleted in this job.
- **Err\_Number:** If the sub job fails, the database produces the error number due to which it fails. Different databases produces different error number and this can vary depend

upon what database is used ETL processing. If there is a failure in the sub job this field provides the information about it.

- **Err\_Procedure:** If the sub job fails, this field provides the name of stored procedure due to which the job failed. In case of multiple stored procedures implemented in one sub job, this field provides information about the one failed.
- **Err\_line:** If the sub job fails, this field provides exactly which line number has the error in the script or the stored procedure.
- **Err\_Message:** This is also a part of the error object. This field tells the error message if the job fails.

### **3.3.2.5 ETL\_JOB\_DEPENDENCY**

In any ETL job system there could be potential dependencies. It could be possible that Job A is dependent on Job B which could be further dependent on Job C. This means unless all the operations are performed by Job A, Job B should not start and unless all the operations are performed by Job B, Job C should not start. This metadata information can be used in a flexible way. Job dependencies are common issues in data warehouse world which can be taken care by creating this table. The following explains the columns level details:

- **Source\_Job\_Id:** This column is a foreign key relationship to the Job\_Id for Job\_Setup table. This is the source job id to which some other job will be dependent on. If Job A is dependent on B then this Source Job is job A.
- **Dependent\_Job\_Id:** This column is a foreign key relationship to the Job\_Id for Job\_Setup table. This is the dependent job id and it is dependent on Source\_Job\_Id. If Job A is dependent on B then this Dependent Job is job B.
- **isActive:** This is a flag to see if this dependency is active or not. There could be some dependencies which you don't want anymore, it is a good idea to keep the history.
- **Start\_Date:** The dependency of the jobs will start at some point of time. This field will be populated when a job dependency relationship is defined at the first time. This is a good practice as it keeps the historical record of the dependencies of ETL.
- **End\_Date:** The dependency could also end at some point of time. This End\_Date will be populated at the time dependency exists no more. In other cases it will 'NULL'.

### **3.3.2.6**

#### **ETL\_JOB\_SOURCE\_DESTINATION\_TABLE\_INFO**

This table provides information about the source tables and destination tables on which it depends for the particular job. For example, if one wants to know which tables are associated with the particular job, this table can provide the information. If source data is

coming as a replicated and there is some lag in the data, it will be very useful from this table to tell which jobs will be affected or which reports will be affected.

So, the current model is very thoughtful and can be applied to any ETL job system. This data model gives the flexibility of controlling the different ETL processes, handling error exceptions, handling the dependency relationships and also can be easily extended for the custom needs.

In the next section ETL job framework and an algorithm is presented. Although this framework can be extended to the custom needs according to the users requirements and depending upon how much the developers want to make the process flexible. The aim of this framework is to present the idea of how to use of this metadata model in ETL.

### **3.4 ETL JOB FRAMEWORK MODEL AND ALGORITHM**

This section presents an ETL job framework and algorithm. This job framework is a very basic framework which makes use of the metadata tables. The idea of presenting this framework is to help understand implementation of the ETL jobs to use the metadata. This framework can be applied to any ETL Job or sub job depending upon the need. This framework will take care of the following things

1. Make sure the source of the data is ready for the ETL job.
2. No multirunning instances of the job.
3. Exception Handling.
4. Logging.
5. Can be easily extended to make sure if the job fails, it restarts from the same location. This logic has not been a part of this framework. This has been done to lower the complexity involved.

The following is the ETL Job framework algorithm :

*Step 1: CHECK IF THE SOURCE DATA IS READY (Get this info by Joining Source\_Objects Table and Job\_Dependency and Object\_Refresh\_Transactions)*

*IF DATA IS NOT READY (NOTIFICATION != 1)*

*THEN 'DO SOME LOGIC '*

*ELSE*

*STEP 2 :SET THE NAME OF THE JOB WHICH IS SAME AS NAME IN JOB\_SETUP TABLE*

```

BEGINTRY
    GET THE JOB_ID FROM JOB NAME
EXECUTE @RC= LOG_JOB_START (JOB_ID,JOB_NAME)
IF @RC=0 --MEANS WE ARE GOOD
    BEGIN
        --'DO THE LOGIC HERE OR CALL THE SUB JOBS'
        EXECUTE @RC1=  -- SUB JOB NAME (@ERROR OUTPUT)
        IF @RC1!=1
        BEGIN
            RAISEERROR(@ERROR)
        END
    ELSE --IF CONTROL REACHES HERE ,IT MEANS JOB GOT SUCCESSFUL
        BEGIN
            SET @ERROR= 'NULL'
            SET @MESSAGE='SUCCESS'
            EXECUTE @RC=LOG_JOB_END(@ERROR,@MESSAGE)
        END
    END
    ELSE --THIS IS A CASE WHEN @RC!=0 , WHICH HANDLES THE CASE IF THE JOB IS
    ALREADY RUNNING
        RAISEERROR('JOB IS ALREADY RUNNING')
    END TRY
    BEGIN CATCH
        IF @RC1=1 --JOB IS FAILED DUE TO THE SUB JOB CALLED .
        BEGIN
            SET @MESSAGE='FAIL'
            EXECUTE @RC= LOG_JOB_END(@ERROR,@MESSAGE)
        END
        ELSE
            SET @MESSAGE ='FAIL'
            @RC= LOG_JOB_END(@ERROR,@MESSAGE)

```

*END CATCH*

*END*

### **3.5 EXPLANATION OF THE ALGORITHM**

Step 1 : The first thing this algorithm does is check if the source of the data which needs to be captured by this ETL is ready or not . The source of the data could be a data file or table or replicated table. The info about the readiness of the data can be found from source metadata tables which we have discussed before. Join the job\_dependency table with source\_objects table with object\_refresh\_transactions to get the latest status of the source data.

If the source notification is not true, in that case do some logic based on this condition. If the source notification is true , it means the logic can be performed.

Step 2 : The next step is where the logging of the job happens and any error handling happens.

The first thing an algorithm does is get the job\_id from name of job. Note that the name of the job needs to be hard coded and should be same as the name of the job you provide on the job\_setup table.

Step 3: After getting job\_id, call a stored procedure which logs the starting of the job in the ETL\_Jobrun\_Info table. It will also check if the job is currently running. This algorithm does not allow multiple instances of this job running at the same time. This stored procedure will return a code 1 if the job is still running else it will return 0.

When @RC =0 then this is a condition in which one can perform the logic or run the sub job.

Step 4: Call another stored procedure which calls the logic for this ETL job or it may be a sub job. Get a return code for this stored procedure which is called as @RC1. If @RC1=1 then RAISE THE ERROR, the @ERROR object is the SQL generated object with the error related information that is passed to the log\_job\_end.

Step 5: If everything goes well ,then the control reaches to the next line where @MESSAGE is set to 'SUCCESS' that means the job was successful without any error . The @Error object will be NULL in this case. The stored procedure LOG\_JOB\_END will be called and it will update the run info in the ETL\_JobRun\_Info table.

Step 6: Next step is when the @RC !=0 in the first place when it was called LOG\_JOB\_START. It means that this job is currently running,so one cannot run this job again. Then the try block will be finished.

Step 7: Now all the exception will be caught by BEGIN CATCH block. The @RC1=1 condition is for the job which we invoked for the logic. If there is an error in the procedure, @error object will catch it and will log into the JOB\_LOG\_END with status ='FAIL'. If there is no error on the sub job procudre, it could be possible that an error occurred on this script as well. For that there is next block of code for error handling and logging on the ETL\_JobRun\_Info table.

This framework presented is a very good example to show one can use source metadata and ETL metadata to get the clear picture of things flowing in the data warehouse. One can potentially extend this concept to a new level by adding dependencies on ETL jobs.

## **CHAPTER 4**

### **IMPLEMENTATION AND TESTS**

#### **4.1 INTRODUCTION**

This section discusses the implementation of the research model presented in chapter 3 along with test data loaded and gives the feel of how the tables and data look like in the database after the implementation. The discussion on the implementation platform for the model will be done and after that results screenshots will be shown to show the test data in the tables.

#### **4.2 IMPLEMENTATION PLATFORM**

The implementation of the model presented in chapter 3 includes number of steps like creating stored procedures, execute sql scripts. For all these steps, Microsoft SQL Server 2008 R2 has been used for the implementation. The tool used for implementation is SSMS (like SQL Server Management Studio). SSMS is a great tool to work and is very intuitive and easy of work with. The database server has 6GB ram with 640GB disk space providing a decent machine for this implementation. The database called 'Abhinav' is used for the implementation of the model.

#### **4.3 TEST AND RESULTS**

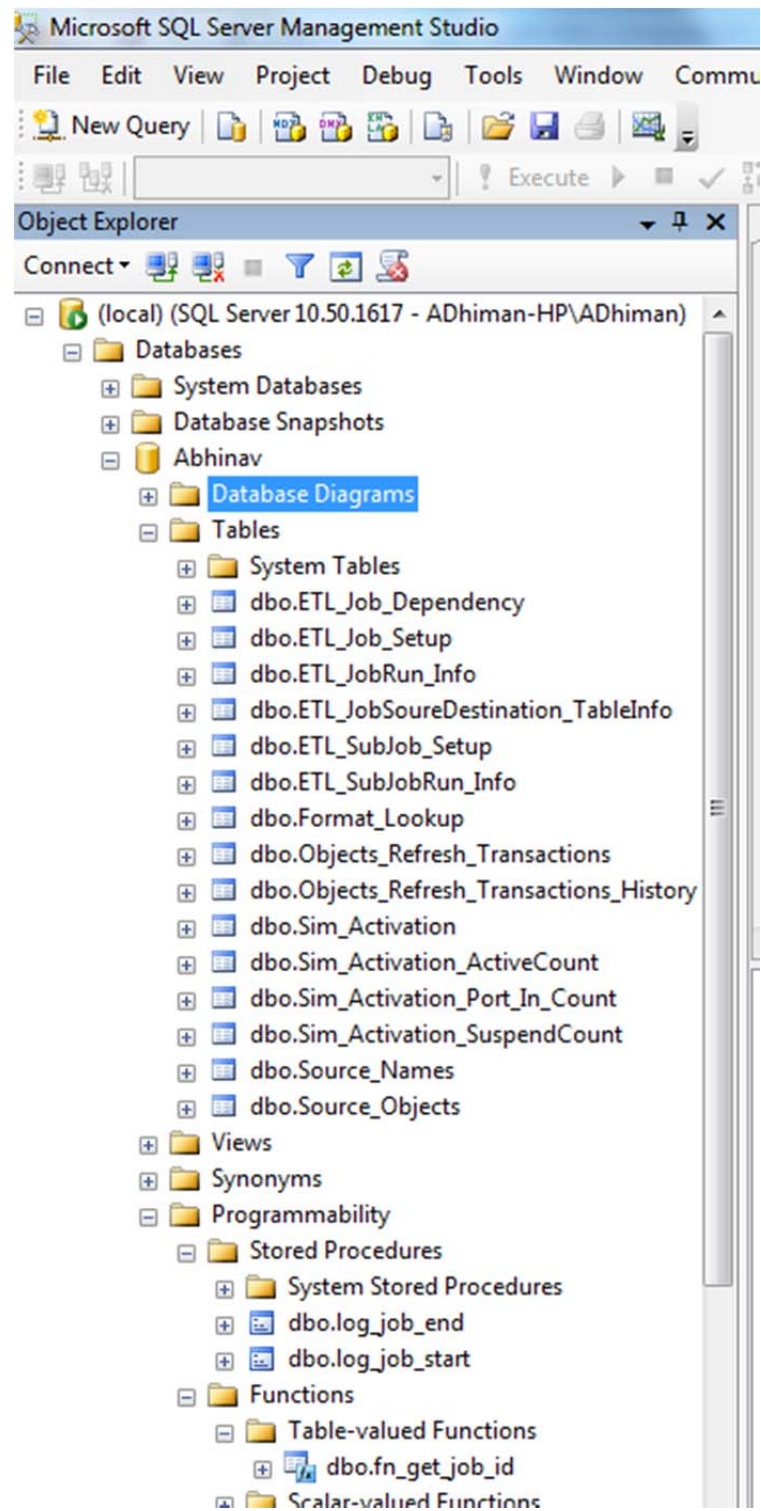
This section presents the result screenshots of the tests being performed on the source metadata and ETL metadata implementation.

##### **4.3.1 Source Metadata Implementation**

Screen Shot 1:

The Figure 4.1 screen shot shows the implementation of the Metadata Model we presented in Chapter 3.

The implementation has been done on SQL Server and name of the database is 'Abhinav'. The Figure 4.1 shot also shows the stored procedures that is a part of the



**Figure 4.1. Results #1.**



framework implementation. The log\_job\_start and log\_job\_end stored procedures and function fn\_get\_job\_id corresponds to naming defined in algorithm in chapter 3.

#### Screen Shot 2:

Figure 4.2 screen shot shows the sample data in Source\_Names table. This table has currently two rows which implies that currently data warehouse have two sources from which data is loaded. This table should be managed at the source metadata level. The isUnderMaintainance flag is for letting the end user know about any delay in the data.

id	name	frequency	format_fk	Source_Mask	IsUnderMaintainance	DataAsOf
1	SQL-01	HOURLY	1	NULL	0	2012-01-29 19:49:10.180
2	Sim_Activation_Active_yyyy_mm_dd.csv	DAILY	2	Sim_Activation_Active_yyyy_mm_dd.csv	0	2012-01-29 20:47:33.603

**Figure 4.2. Results #2.**

#### Screen Shot 3:

Figure 4.3 screen shot shows the Source\_Objects table information. The name of this table helps to keep the naming convention generic in nature. For instance, in most cases different objects like SQL tables, flat files, excel file can act as potential sources. This table shows two source\_objects each belongs to their own source by Source\_id column.

#### Screen Shot 4:

Figure 4.4 screen shot shows the look up table that contains the different formats for the sources. Multiple formats can exist and is always a good idea to create a lookup table to avoid adding junk data in the database.

#### Screen Shot 5:

Figure 4.5 screen shot shows the test data in Object\_Refresh\_Transactions and is a transactional log of the source meta data when the data is ready on the source systems to pull. This table has primary key as id and have a source\_objects\_id which that gives information about which source has the latest data available. The notification flag 1 tells

SQLQuery97.sql - ...-NF (Abhinav) (01) | SQLQuery99.sql - ...-NF (Abhinav) (06) | SQLQuery94.sql - ...

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [id]
    , [Source_id]
    , [Object_name]
    , [object_Description]
FROM [Abhinav].[dbo].[Source_Objects]

```

Results Messages

	id	Source_id	Object_name	object_Description
1	1	1	SQL-01.DBO.Sim_Activation	This object is a table from SQL-01 database
2	2	2	Sim_Activation_Active_yyyy_mm_dd.csv	Daily Active subscribers file

Figure 4.3. Results #3.

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [id]
    , [Format_Name]
    , [Description]
FROM [Abhinav].[dbo].[Format_Lookup]

```

Results Messages

	id	Format_Name	Description
1	1	SqlServer-Database	This format is contains all the data from SQL Server 2008R2 Database
2	2	FlatFile in csv format	This format supports the comma seperated flat file

Figure 4.4. Results #4.

```

SELECT TOP 1000 [id]
    , [source_Objects_id]
    , [DataAsOf]
    , [NotificationFlag]
    , [RecordsCount]
FROM [Abhinav].[dbo].[Objects Refresh Transactions]

```

Results Messages

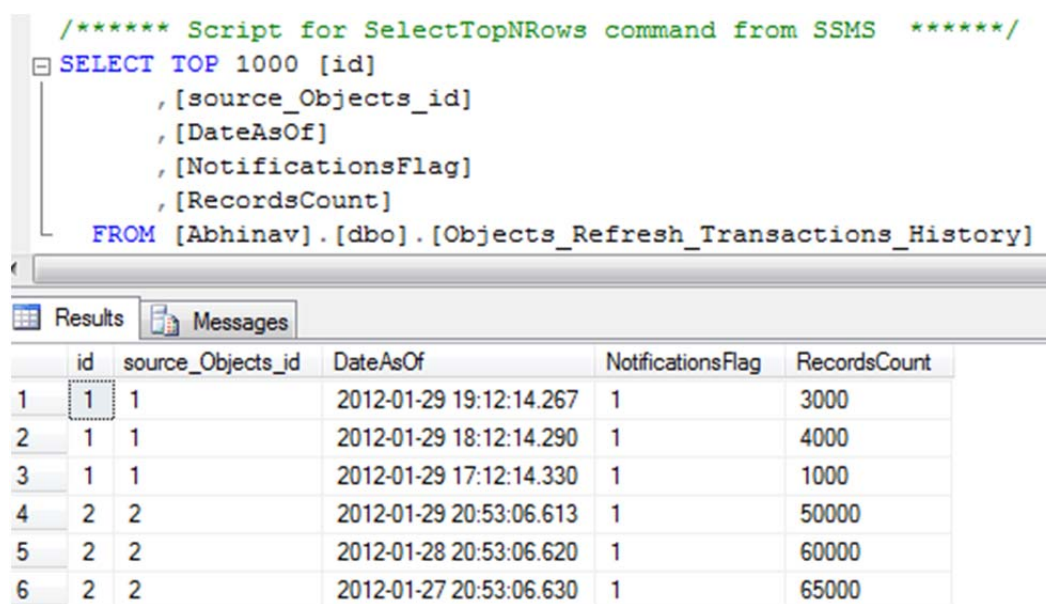
	id	source_Objects_id	DataAsOf	NotificationFlag	RecordsCount
	1	1	2012-01-29 20:10:22.483	1	2000
	2	2	2012-01-29 19:49:10.180	1	70000

Figure 4.5. Results #5.

that the data is ready to be pulled and it also provides the number of records it contains. Note that this table only has the most current information for the source, the historical data gets moved to the `Object_Refresh_Transactions_History` table.

Screen Shot 6 :

Figure 4.6 screen shot shows the `Object_Refresh_Transactions_History` table that contains the historical transactional log for the source metadata. Everytime a new row is inserted in `Object_Refresh_Transactions` table the old one should move to `Object_Refresh_Transactions_History` table.



```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [id]
      , [source_Objects_id]
      , [DateAsOf]
      , [NotificationsFlag]
      , [RecordsCount]
FROM [Abhinav].[dbo].[Objects_Refresh_Transactions_History]

```

	id	source_Objects_id	DateAsOf	NotificationsFlag	RecordsCount
1	1	1	2012-01-29 19:12:14.267	1	3000
2	1	1	2012-01-29 18:12:14.290	1	4000
3	1	1	2012-01-29 17:12:14.330	1	1000
4	2	2	2012-01-29 20:53:06.613	1	50000
5	2	2	2012-01-28 20:53:06.620	1	60000
6	2	2	2012-01-27 20:53:06.630	1	65000

**Figure 4.6. Results #6.**

### 4.3.2 ETL Metadata Implementation

The following screen shots are related to the ETL metadata model implementation. The screen shot shows the test data in the tables and shows how the sample data looks like in database. The data warehouse team can implement this model in number of different way depending upon the requirements. The data model provides the flexibility to extend and implement in numerous ways.

Screen Shot 1:

Figure 4.7 screen shot shows the `ETL_Job_Setup` table. This table is set up once when the new job needs to implemented in the data warehouse.

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [Job_Id]
, [Job_Name]
, [Short_Description]
, [Description]
, [Frequency]
, [Scheduled_Time]
, [Avg_Runtime_Minutes]
, [Max_Runtime_Mins]
, [Min_Runtime_Mins]
FROM [Abhinav].[dbo].[ETL_Job_Setup]

```

Job_Id	Job_Name	Short_Description	Description	Frequency	Scheduled_Time	Avg_Runtime_Minutes	Max_Runtime_Mins	Min_Runtime_Mins
1	GetActiveSubscribers	calculates the active subscribers in the company	This job calculates the active subscriber	DAILY	00:00:00.000	2	10	NULL
2	HighLeveSA_Metrics	Provides High Level SA Metrics	Active,Port-In and Suspends	DAILY	01:00:00.000	3	7	3

**Figure 4.7. Results #7.**

Screen Shot 2:

Figure 4.8 screen shot shows the ETL\_SubJob\_Setup table. The assumption has been made that every job have one or more steps that needs to be implemented. The information about the sub job is provided in this table. This table is also setup once, when the new job needs to implemented in the data warehouse.

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [Sub_Job_Id]
, [Job_Id]
, [Sub_Job_Name]
, [Short_Description]
, [Description]
, [Avg_Runtime_Minutes]
, [Max_Runtime_Mins]
, [Min_Runtime_Mins]
, [Load_Frequency]
FROM [Abhinav].[dbo].[ETL_SubJob_Setup]

```

Sub_Job_Id	Job_Id	Sub_Job_Name	Short_Description	Description	Avg_Runtime_Minutes	Max_Runtime_Mins	Min_Runtime_Mins	Load_Frequency
1	1	UpdateSuspendRecords	Updates Suspend to Deactivate	Updates Suspend to Deactivate	2	10	NULL	DAILY
2	1	InsertActiveRecords	insert active subs to new table	insert active subs to new table	2	10	NULL	DAILY
3	2	InsertMetrics	Insert Metrics	Insert Metrics	3	7	3	DAILY

**Figure 4.8. Results #8.**

Screen Shot 3:

Figure 4.9 screen shot shows the ETL\_JobRun\_Info. This table have the information about the ETL jobs which are currently running, completed and failed.

Screen Shot 4:

Figure 4.10 screenshot shows the execution of the log\_job\_start stored procedure along with providing the job\_name parameter. This stored procedure does not support running its multiple instances and will have the status in ETL\_JobRun\_Info table as

Script for SelectTopNRows command from SSMS

```

SELECT TOP 1000 [RunId]
, [Job_Id]
, [Start_Time]
, [End_Time]
, [Elapsed_Time]
, [Run_Status]
, [Run_By]
, [Extract_Count]
, [Load_Count]
, [Error_Step]
, [LastRunStatus]
FROM [Abhinav].[dbo].[ETL_JobRun_Info]

```

RunId	Job_Id	Start_Time	End_Time	Elapsed_Time	Run_Status	Run_By	Extract_Count	Load_Count	Error_Step	LastRunStatus
1	2	2012-01-30 22:14:47.050	2012-01-30 22:18:00.587	NULL	SUCCESS	ADhiman-HP\ADhiman	NULL	NULL	NULL	NULL

**Figure 4.9. Results #9.**

```

DECLARE @RC int
DECLARE @job_id int
DECLARE @job_name varchar(100)

-- TODO: Set parameter values here.
set @job_id=2
set @job_name='HighLeveSA_Metrics'
EXECUTE @RC = [Abhinav].[dbo].[log_job_start]
    @job_id
    ,@job_name
GO

```

Messages

**RUNNING**

**Figure 4.10. Results #10.**

'RUNNING' if run multiple times more than one , meaning that the job is running.If one try to run this procedure again then the procedure will check to see if the job is currently running, if thats the case then it won't let it run since the job is currently running. This logic is done at this procedure level and is also discussed in chapter 3 when framework algorithm was presented.

Screen Shot 5:

Figure 4.11 screenshot shows the information in the ETL\_SubJobRun\_Info table. This table has information about the steps implemented by the ETL Job. For the run\_id =1 status is 'SUCCESS' for this sub step.

The screenshot shows a SQL query window with the following text:

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [SubJobRunId]
, [Sub_Job_Id]
, [Start_Time]
, [End_Time]
, [Elapsed_Time]
, [Run_Status]
, [Run_Id]
, [No_OfRowsInserted]
, [No_OfRowsUpdated]
, [No_OfRowsDeleted]
, [Err_Number]
, [Err_Procedure]
, [Err_Line]
, [Err_Message]
FROM [Abhinav].[dbo].[ETL_SubJobRun_Info]

```

The Results pane shows the following data:

SubJobRunId	Sub_Job_Id	Start_Time	End_Time	Elapsed_Time	Run_Status	Run_Id	No_OfRowsInserted	No_OfRowsUpdated	No_OfRowsDeleted	Err_Number	Err_Procedure	Err_Line	Err_Message
1	3	2012-01-30 22:21:48.720	2012-01-30 22:31:48.720	10	SUCCESS	1	20	NULL	NULL	NULL	NULL	NULL	NULL

**Figure 4.11. Results #11.**

Screen shot 6:

Figure 4.12 screenshot shows the dependency table data that can be used in ETL in maintaining the dependencies between data warehouse objects. The data shows that job\_id 1 is dependent on job id 2. So one can implement some logic based on this dependency. Large number of dependencies can be managed effectively using this table.

The screenshot shows a SQL query window with the following text:

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [Source_Job_Id]
, [Dependent_Job_Id]
, [IsActive]
, [Start_Date]
, [End_Date]
FROM [Abhinav].[dbo].[ETL_Job_Dependency]

```

The Results pane shows the following data:

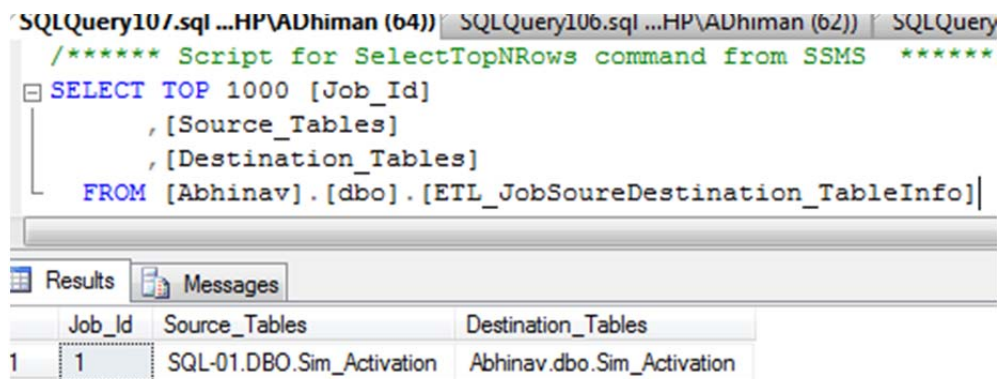
Source_Job_Id	Dependent_Job_Id	IsActive	Start_Date	End_Date
1	2	1	2012-01-29 21:06:48.047	NULL

**Figure 4.12. Results #12.**

Screen Shot 7:

This table provides relationship for the source objects and the destination objects (see Figure 4.13). These objects refer to the tables in source\_objects table which means that the data in the destination table is extracted from which source. This information is very helpful in providing the dependencies between tables and to let the end user know about the delay in the data if there is any.





**Figure 4.13. Results #13.**

So we have come to an end of the implementation phase of the data model. The implementation is done for the source metadata model and ETL metadata model.

#### 4.4 THESIS HIGH LEVEL SUMMARY

This section is being added to help understand the high level flow of the data warehousing and the importance of metadata model during the data warehouse design and implementation. This discussion is being supported by the real life examples, which helps explain the whole picture.

Let's say you are an executive of a company, which sells cell phones. The sales of cell phone occurs online and in stores. As an executive you would want to know how the sales are doing every day. You may also want to know which store is doing well and which store is not, so that you know where your business is lacking or doing well. There can be thousands of such questions which management team needs answers to make business decisions.

Typically, the computer systems on which all of sales/transactions occurs are on different systems. One of them could be a mainframe system; another could be a web application and so on. As a result, similar data resides on different systems. As an executive you would want to see the whole picture of your business. The whole picture can only be seen if the data from different sources can combined in such a way that business analysis can be done. The combining of the data is being done by a system known as data warehouse and the process by which the data is combined is known as data warehousing. This leads to creating a data warehousing for business analysis.

Now as a developer and as a data warehouse designer, they need to know the data in the different systems so that data can be combined and make sense. For the data warehouse,

the first system, which developers of data warehouse have to interact, is the source of the data. In this example the source of the data is the data coming from the systems in stores and the data coming from the online source. Analyzing the source systems becomes the first component of data warehouse because the data warehouse architects and developers has to understand the business. This is the reason for the source systems being the part of data warehouse architecture.

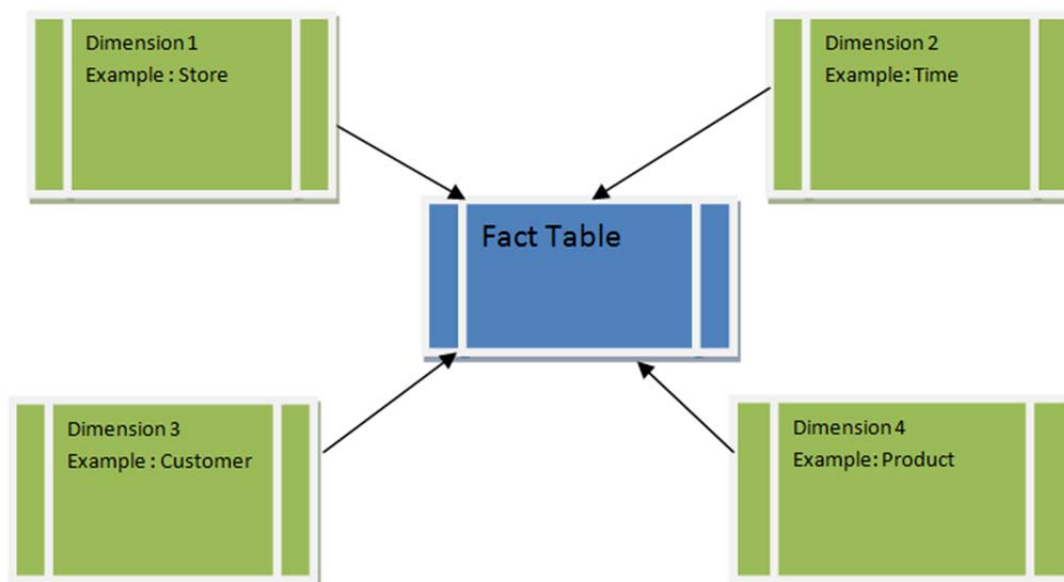
Once the source data is being understood, the architects and developers must create a process by which they will extract the data from the different sources (in this example: Online Source and Stores), clean it (handling bad data rows, exception handling) , do the transformations (data type conversions, data casting rules), and load into a data model (star schema). This whole process takes more than 70% of the effort in data warehousing and it is the most important. This whole process is also known as ETL process, which is extraction, transformation and load. Each ETL process can do some task and it may or may not be dependent on other ETL process.

So once we have the data from the source, the ETL process will fetch the data, clean it and will load it into a data model from which the querying and analysis can be done. This is the reason why the middle layer of data warehousing is called as ETL processing. The data model into which the cleaned, transformed data will be loaded is star schema or snowflake schema. Star schema has a fact table and dimensions table. Fact table contains the numeric measures by which you want to measure your business. Example: In our scenario, a total sale in a store A is numeric measure and fact table will have this numeric numbers. Also the fact table can have data related to money, like total amount of sales though online shopping cart. A fact table has a numeric measures and the foreign keys to the dimensions in its each data row.

Dimensions table contains the data by which you want to measure your business against. These are by which you want to slice and dice the data. Dimension tables are 'by' of the business. Example: By Date, By Geography, By Product. In our scenario, an executive would want to see total sales of Samsung phones in last quarter or total sales of all cell phones yesterday, or total sales of blackberry phones in California in the last year. The information in the fact table (numeric measure such as sales) can be sliced and diced by the dimensions (By date, product or geography). The star schema is a Ralph Kimball's model and



is a standard way of implementing the data modeling in data warehouses. We store the data in star schema to slice and dice the information coming from source for reporting and data analysis. Architecture of typical star schema is given in Figure 4.14 (seen previously in this thesis).



**Figure 4.14. Star schema example.**

After this the data in the star schema can be used by the presentation layer to query it and create reports or create a data cube or create a executive dashboards. The presentation layer will get the data from the star schema and will be presented in any of the forms said above. This layer is usually decoupled from the other components of data warehouse and is visible to only end user.

Now during all the phases of data warehousing, there is one element which is always there is always hidden i.e. Metadata. Metadata is nothing but data about the data. In data warehousing, there is so much data to handle that usually developers and architects tend to ignore the metadata involved with data warehouse. Metadata is the information about the implementation of data warehousing and it needs to be capture at every phase of data warehouse to provide end to end integration, help monitoring the data warehouse processes, controlling the data warehouse processes, help creating process automation.

Metadata is attached to the source systems, which is called as source system metadata. Source system metadata includes information like Source schemas, Data Format,

Relational source system tables and DDL, Update frequencies of original sources, Access rights privileges, roles, Extraction of data, Job schedules. To capture the source metadata information the thesis has provided a standard way of putting the data into a source system metadata model.

Metadata is also attached to the ETL layer of data warehouse. ETL metadata includes information like Definitions of facts and dimensions tables, Job details for sources and look up attributes, Data cleaning rules, Mapping information, Required Transformations, Target schema designs, source to target data flows, Audit records, ETL job run time logs, summaries, and time stamps, Business descriptions of extract processing etc. The most important part of data warehousing is the ETL stage and there is a lot of metadata that needs to be captured. This thesis provides a standard way of implementing a ETL metadata model which will capture all the information mentioned above for the data warehouse processes.

Suddenly the task of managing the metadata seems equally challenging as creating a data warehouse itself. This is true and metadata needs to be generated, stored and should be used for the success of data warehouse. So this way by having the metadata around data warehouse one can control the processes, manage it and monitor it. This is the biggest benefit of following metadata approach.

## **CHAPTER 5**

### **RESEARCH CONTRIBUTIONS AND POSSIBLE FUTURE WORK**

This chapter summarizes the research contributions of this thesis and identifies the possible future scope.

#### **5.1 RESEARCH CONTRIBUTIONS**

The research contributions of this thesis are given below.

1. The literature about the data warehousing concepts is reviewed .An in depth analysis of topics such as difference between OLTP and OLAP systems and identifying the myths about data warehousing is discussed.
2. Then, the architecture of a typical data warehouse is explained and also presented the flow of processes of every phase in data warehousing. Moreover, the OLAP cube technology is reviewed along with dimensional modeling techniques like star schema and snow flake schema.
3. Also, the value of metadata in the data warehouse development is identified and explained in depth the concept behind the metadata and the way metadata is supposed to be classified and how it can help in taking data warehousing to another level.
4. The research also presents the metadata model which is divided into two parts : first is source metadata and second for ETL process metadata. This data model is platform independent and has capability to capture different states of data to help creating the cleaner approach for data warehousing.
5. An approach to implement the ETL job is presented by providing a job framework algorithm that shows how the metadata models helps to create flexible data warehouse processes. The fact that one can monitor the data warehouse activities and control the activities makes this model a unique one. Everything in this methodology is made from scratch, so one can easily customize to any level.
6. The implementation is done in SQL Server 2008 R2 server and also presented the screen shots with the test data. The implementation of this model presented how the data will look like and what feature it can provide.
7. The test results and the implementation of the research model shows the tip of the iceberg of what we can be done with this approach. Using this model, one can add many more functionalities to the data warehouse process to control it, to manage it, and to monitor it.

To summarize, the metadata has always been considered unimportant in the data warehouse world but most people realize its importance afterwards. So, the main focus of this research is to make a standard metadata implementation model that one can use. The idea is to make sure the implementation of metadata is done at the time data warehouse is being designed to get the benefits it can provide and they are indeed countless.

## **5.2 FUTURE WORK**

The research contribution can be enhanced by presenting the third metadata model for reporting which is also known as front room metadata. The front room metadata can make use of the ETL metadata and source level metadata to do logic on the front room side that will help in implementing the metadata model end to end. [15]

As already been discussed, this model is just the tip of the iceberg and there is so many ways the research can be extended. It can be extended to implement the metadata model in real time data warehousing where the source is constantly providing data every second.

## REFERENCES

- [1] S. MARCH, *Special issue on heterogeneous distributed database systems*, IEEE Computer, 24 (1991), pp. 183-236.
- [2] G. WIEDERHOLD, *Mediators in the architecture of future information systems*, IEEE Computer, 25 (1992), pp. 38-49.
- [3] J. WIDOM, *Proceedings of the 4<sup>th</sup> International Conference on Information and Knowledge Management (CIKM)*, University of Maryland, Baltimore, 1995.
- [4] C. SHILAKES AND J. TYLMAN, *Enterprise information portals*. Enterprise Software Team, [http://ikt.hia.no/perrep/eip\\_ind.pdf](http://ikt.hia.no/perrep/eip_ind.pdf), accessed September 2011, last modified November 1998.
- [5] B. INMON, *The data warehouse budget*. DM Review Magazine, <http://www.datawarehouse.inf.br/Papers/inmon%20budget-1.pdf>, accessed October 2011, n.d.
- [6] R. KIMBALL, *A dimensional modeling manifesto*, J. Data Base Mgmt. Syst., 10 (1997), pp. 58-70.
- [7] R. KIMBALL, L. REEVES, M. ROSS, AND W. THORNTHWAITE, *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*, John Wiley & Sons, Hoboken, New Jersey, 1998.
- [8] R. KIMBALL AND J. CASTERTA. *The Data Warehouse ETL Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*, John Wiley & Sons, Hoboken, New Jersey, 2004.
- [9] A. SIMITSIS, P. VASSILIADIS, AND T. K. SELLIS, *Proceeding of the 21<sup>st</sup> International Conference on Data Engineering*, IEEE Computer Society, Tokyo, Japan, 2005.
- [10] P. VASSILIADIS, A. SIMITSIS, P. GEORGANTAS, M. TERROVITIS, AND S. SKIADOPOULOS, *A generic and customizable framework for the design of ETL scenarios*, J. Inform. Syst., 30 (2005), pp. 492-525.
- [11] 1KEYDATA, *Conceptual data model*. 1keydata, <http://www.1keydata.com/datawarehousing/conceptual-data-model.html>, accessed January 2012, n.d.
- [12] P. SUMA, *Logical modeling of ETL processes using XML*. OhioLINK ETD Center, <http://etd.ohiolink.edu/send-pdf.cgi/P%20Snehalatha%20Suma.pdf?ucin1276948742>, accessed December 2011, n.d.
- [13] K. KUMAR, *Role of metadata in data warehousing environment*, Master's thesis, Luleå University of Technology, Luleå, Sweden, 2006.
- [14] T. HAMMERGREN AND A. SIMON, *Data Warehousing for Dummies*, John Wiley & Sons, Hoboken, New Jersey, 1997.

- [15] DATAWAREHOUSE4U, *OLTP* vs. *OLAP*. Datawarehouse4u,  
<http://datawarehouse4u.info/OLTP-vs-OLAP.html>, accessed December 2011, n.d.