Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Diploma Thesis

Data Warehouse Design and Implementation

Jakub Matuška

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Jakub Matuška

Economics and Management

Thesis title

Data Warehouse Design and Implementation

Objectives of thesis

This diploma thesis investigates data warehousing concepts. The main objective of this thesis is to examine design and implementation process of the data warehouse solutions in a selected company and to assess benefits, costs, and feasibility of the selected solutions. The partial goals of the thesis are:

- To introduce the theoretical framework of data warehousing, architecture and implementation procedures of a data warehouse project.

- To identify and examine the selected company, its business requirements and processes.

- To analyze and evaluate the selected solutions and to produce further recommendations.

Methodology

To accomplish the stated objectives, methods of deduction, induction, synthesis and extraction are being used. The theoretical part deals with the literature review of data warehousing. The practical part is focused on the implementation process and selection of the optimal data warehouse solution in a given company using multi-criteria decision analysis (MCDA) methods. Feasibility of this solution will be assessed. Based on the literature review and outcomes of the practical part, final conclusions will be formulated.

The proposed extent of the thesis

60 – 80 pages

Keywords

Data Warehouse, Business Intelligence, Operational Data Store, Data Mart, ETL, Design, Implementation, Benefits, Costs, Solution, Architecture, Delivery, Evaluation

---

Recommended information sources

HUGHES, Ralph. Agile Data Warehousing for the Enterprise: A Guide for Solution Architects and Project Leaders. Morgan Kaufmann, 2015. ISBN 978-0123964649.

INMON, William H. Data architecture: a primer for the data scientist. Waltham, MA: Elsevier, 2014. ISBN 9780128020449.

KIMBALL, Ralph a Margy ROSS. Data warehouse toolkit: the definitive guide to dimensional modeling. Third edition. Indianapolis: John Wiley & Sons, 2013. ISBN 978-1-118-53080-1.

SHERMAN, Rick. Business intelligence guidebook: from data integration to analytics. Amsterdam: Elsevier, Morgan Kaufmann is an imprint of Elsevier, 2015. ISBN 9780124114616.

Expected date of thesis defence

2016/17 WS – FEM

The Diploma Thesis Supervisor

Ing. Miloš Ulman, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 18. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 28. 11. 2016

Declaration

I declare that I have worked on my diploma thesis titled "Data Warehouse Design and Implementation" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 30/11/2016 _____

Jakub Matuška

Acknowledgement

I would like to thank Ing. Miloš Ulman, Ph.D. for his advice and support during my work on this thesis.

Data Warehouse Design and Implementation

Summary

The thesis “Data Warehouse Design and Implementation” deals with the implementation process and design of a data warehouse in the banking sector. This thesis is divided into two main sections, theoretical and analytical. The theoretical part introduces essential terms and methods related to data warehousing. It focuses primarily on the data warehouse architecture, design and implementation process, and system development life cycle. The analytical section applies the framework analyzed in the theoretical part. It focuses on the introduction of the bank in which the data warehouse is implemented, identification of the particular business case, data warehouse implementation process, and the selection of the optimum database management software for the selected bank. The optimum database management software is selected using Multi-Criteria Decision Analysis. After the analytical section is finished, results and findings are presented and developed. Finally, there is the conclusion section that summarizes and reflects on the theoretical, analytical and result parts.

Keywords: Data Warehouse, Business Intelligence, Operational Data Store, Data Mart, ETL, Design, Implementation, Benefits, Costs, Solution, Architecture, Delivery, Evaluation, Bank, Project

Design a Implementace Datového Skladu

Souhrn

Tato diplomová práce s názvem „Design a Implementace Datového Skladu“ se zabývá procesem vývoje a implementace datového skladu v bankovním sektoru. Práce je rozdělena na dvě hlavní sekce, teoretickou a analytickou. Teoretická část představuje zásadní termíny a metody problematiky datových skladů. Zaměřuje se zejména na architekturu datových skladů, proces návrhu a implementace datového skladu, a životní cyklus informačního systému. Analytická část aplikuje teoretická východiska představená v teoretické části. Hlavní cíl praktické části je představit banku a její případ, analyzovat proces implementace datového skladu a vybrat optimální databázový systém pro tuto banku. Optimální databázový systém je vybrán s použitím metody vícekriteriální hodnocení variant. Analytickou část následuje zhodnocení výsledků a prezentace nových poznatků. V závěrečné části je zhodnocení průběhu veškerých předcházejících částí a jejich vyhodnocení.

Klíčová slova: Datový sklad, Business Intelligence, Operational Data Store, Data Mart, ETL, Design, Implementace, Náklady, Výhody, Řešení, Architektura, Dodání, Vyhodnocení, Banka, Projekt

Table of Content

List of Figures

List of Tables

1 Introduction

It was in the second half of the 20th century when the Digital Revolution broke out. As a result of this Revolution, the shift from the mechanical and analogue electronic technology to the digital electronics has begun. In other words, the era of digital computers, phones and the internet started. These new technologies had profound impact on everybody in the developed world. New technologies came up with new means of communication and influenced the way in which common people and companies function. Communication became faster and companies more efficient. Nowadays, the world is becoming even more interconnected, more mobile and more sharing than it was a couple of years ago which caused volumes of data to grow. One does not have to go very far to make certain of that. For example, in 2015, 90% of the world data content had been generated in the past two years (Iteuropa, 2015). This growth of data amounts increase demand for the analysis of data because many companies believe that they can take advantage of better business insights. It is very important to realize that the growing amount of sources generating data make the analysis very complicated because the data from these sources usually lack integration.

The data warehouse is an environment that integrates data and allows the analysis of data to get the relevant and better information for decision-making. This industry has been gaining an increasing amount of relevance due the previously mentioned data amounts growth and increase in the demand for data analysis. This is why this thesis deals with the data warehouse design and implementation. It consists of two main parts, theoretical and analytical. Theoretical part explores the key concepts related to the data warehousing which are after applied on the particular case of the data warehouse implementation in the banking sector. Results and finding acquired during the implementation process are then presented to better

2 Objectives and Methodology

2.1 Objectives

This diploma thesis aims to investigate data warehousing concepts from the theoretical and practical perspective applied in the banking sector. Objectives of the literature review are to introduce theoretical framework of data warehousing with emphasis on the data warehouse architecture and the implementation process related to a data warehouse project. As far as the analytical part is concerned, its main objectives are as follows. The first goal is to identify the particular case of the selected company interested in the data warehouse solution. The second goal is to find the optimum data warehouse solution taking into consideration key business needs and processes of the selected company as well as costs, benefits, and feasibility of the considered solutions. The last goal is to perform a thorough analysis of the data warehouse design and implementation process.

2.2 Methodology

In order to achieve the stated objectives, it is essential to apply methods of deduction, induction, synthesis and extraction. In order to complete the literature review, it is needed to collect high quality and relevant data warehousing sources. It is very important to focus particularly on the content that is the most relevant for the practical part. The analytical part is conducted in a similar manner. Thorough understanding of a company's business processes needs and processes is necessary. After, theoretical framework and analytical techniques such as multi-criteria decision analysis (MCDA) are applied in order to reach the stated goals. Finally, based on the literature review and outcomes of the analytical part, final conclusions are drawn and recommendations provided.

3 Theoretical Part

The intent of this part is to introduce important concepts related to data warehousing considering the limited scope of this thesis.

3.1 Terminology

3.1.1 Data

Data is usually defined as unprocessed facts and figures without any interpretation or analysis (Elearn, 2013). Other sources define it as discrete, objective facts or observations, which are unorganized and unprocessed and therefore have no meaning or value because of lack of context and interpretation (Rowley and Hartley, 2008). Since the term data is closely related to the concept of information and knowledge, it is better to understand it using examples.

3.1.2 Information

Information is very often mistaken for data, however, there is a difference. Literature says that information is data that has been interpreted, and therefore has a meaning for someone or something (Elearn, 2013). Alternatively, it can be defined as refined data that has become useful for some form of analysis (Kelley, 2002). Another very good definition is that information is data that have been recorded, classified, organized, related, or interpreted within a framework so that meaning emerges (Lanning, 2014). As it can be seen, these definitions have a common point. Information is basically processed data with meaning.

3.1.3 Knowledge

Knowledge with reference to the concept of data and information is defined as a combination of information, experience and insight that may positively impact the individual or the organization (Elearn, 2013). Perhaps the most complex definition that also serves as a foundation for other definitions is that knowledge is a mix of framed experience, values, contextual information, expert insight and intuition that provides a framework and environment for evaluation and incorporation of new experience and

information (Wallace, 2007). A short but sufficient definition of knowledge is that knowledge is simply information combined with understanding and capability (Rowley and Hartley, 2008).

3.1.4 Understanding the difference

Data, information, and knowledge are closely related concepts whose boundaries are sometimes very unclear. The main factor that differentiates them is the level of processing. Data are just raw facts and figures that becomes information with interpretation or application for purpose. Information becomes knowledge when some sort of experience or insight is provided. To better understand the difference, figure 1 and an example are provided.

3.1.5 Big Data

Big Data is a loosely defined term used to describe data sets so large and complex that they become awkward to work with using standard statistical and relational database software. The rise of digital and mobile communication has made the world become more connected, networked, and traceable and has typically lead to the availability of such large scale data sets (Raine and Wellman, 2012). Big data can be characterized by 4Vs: Volume, Velocity, Variety, and Value.

3.1.5.1 Volume

Volume stands for high amounts of low density data, that is, data which is of unknown value, such as social networks feeds, clicks on a web page, network traffic, sensors capturing data at speed of light, and many others. It is the task of big data to convert low density data into high density data, that is, data that has value. For some companies, it may be tens of terabytes, for other hundreds of petabytes.

3.1.5.2 Velocity

In this context, velocity refers to the fast rate that data is received and perhaps acted upon. In other words, it is the speed at which data is generated and processed to meet demands

and challenges. The highest velocity data normally streams directly into memory versus being written to disk.

3.1.5.3 Variety

Variety represents new unstructured data types. Unstructured and semi-structured data types, such as text, audio, and video require additional processing to both derive meaning and the supporting metadata.

3.1.5.4 Value

Data has intrinsic value—but it must be discovered. There is a range of quantitative and investigative techniques to derive value from data – from discovering a consumer preference or sentiment, to making a relevant offer by location, or for identifying a piece of equipment that is about to fail. The technological breakthrough is that the cost of data storage and compute has exponentially decreased, thus providing an abundance of data from which statistical sampling and other techniques become relevant, and meaning can be derived. However, finding value also requires new discovery processes involving clever and insightful analysts, business users, and executives. The real Big Data challenge is a human one, which is learning to ask the right questions, recognizing patterns, making informed assumptions, and predicting behavior (Oracle 2016).

3.1.6 Business Intelligence

Business intelligence is a term that that is commonly associated with data warehousing. It can be defined and interpreted in different ways, but in the core of every definition there is a mention that business intelligence deals with transformation of data to useful information to make better decisions. The first definition of business intelligence was coined in 1958 when IBM researcher Hans Peter Luhn defined business intelligence the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal (Cebotarean, 2011). Business intelligence is currently defined as: “An umbrella term that includes the applications, infrastructure and tools, and best practices that enable access to and analysis of information to improve and optimize (Gartner, 2016).” Besides, it can also be defined in a following way: “Business intelligence is an active, model-based, and prospective approach to discover and explain hidden, decision-

relevant aspects in large amounts of business data to better inform business decision processes (Liebowitz, 2006).” To paraphrase the previous definition and better understand the meaning, business intelligence is simply set of tools and techniques that are employed in data collection, analysis, and improved decision making.

When talking about the distinguishing factors of business intelligence and data warehousing, business intelligence can be interpreted either as a top layer of data warehousing architectural solution such as dashboards and analytics, or as an umbrella term for all relevant processes and technologies including data integration, master data management, data warehousing, performance management, reporting, analytics and dashboards (Cebotarean, 2011).

3.1.7 Business Intelligence vs. Big Data

There is a difference between business intelligence and big data. Business intelligence applies descriptive statistics with data with high information density to reach goals. On the contrary, big data analyzes big amounts of data with low information density applying inductive statistics and nonlinear system identification. Similarly, big data tries to explore and analyze relationships, dependencies, trends, and perform predictions of outcomes and behaviors.

3.1.8 Database

A database can be defined as an organized collection of data used for the purpose of modeling some type of organization or organizational process. It does not matter what kind of tool or software is used for gathering and storing the data. As long as the data are gathered in an organized manner for a specific purpose, it is a database (Hernandez, 2003). Another definition is that a database is a collection of many different types of interrelated stored data that serves the needs of multiple users within one or more organizations (Teorey, 2011).

3.1.9 Database Management System (DBMS)

A generalized software system for storing and manipulating databases is called Database Management System (Lightstone, 2007). It provides its users and developers with a systematic way to create, retrieve, update and manage data. The database management

software is simply an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible. There are many types of database management systems such as columnar, in-memory or NoSQL (Database Management System, 2016), however, the most popular is the relational database management system (Lightstone, 2007). The relational database management systems are the focus of this thesis.

3.1.10 SQL

Structured Query Language (SQL) is a standardized special-purpose programming language used for managing relational databases and performing various operations on the data in them. It was created in the 1970 and it has been used by database administrators and developers ever since. It is called a query language, however, its possibilities are not limited only to simple data retrieval (Haan et al., 2014). There are 16 primary SQL commands separated into the following groups:

The Data Manipulation Language (DML) commands:

- SELECT
- INSERT
- UPDATE
- DELETE
- MERGE

The Data Definition Language (DDL) commands:

- CREATE
- ALTER
- DROP
- RENAME, TRUNCATE,
- COMMENT

The Data Control Language (DCL) commands:

- GRANT
- REVOKE

The Transaction Control Languages (TCL) commands:

- COMMIT

- ROLLBACK
- SAVEPOINT (Ramklass 2014)

3.2 Data Warehousing Concepts

3.2.1 Data Warehouse

The data warehouse (DW) can be defined as a database designed to enable business intelligence functionality. It is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from the transaction data, but also other internal or external sources. The data warehouse separates analysis workload and transaction workload which helps in maintaining historical records and analyzing data. However, the data warehouse is not only a database. The data warehouse environment also includes source systems, extraction-transformation-load (ETL) system, and BI solutions (Lane and Potineni, 2014). However it is not the only nor the ultimate definition of the DW since it can also be defined as a system that retrieves and consolidates data periodically from the source systems into a dimensional or normalized data store. It usually keeps years of history and is queried for business intelligence or other analytical activities (Rainardi, 2008). Linstedt provided a definition that the data warehouse is a data-driven decision support system that supports the decision-making process in a strategic sense and, in addition, operational decision-making (Linstedt and Olschimke, 2016). Bill Inmon, the father of data warehousing (Franks, 2013), defined the data warehouse as a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. The listed attributes stand for:

Subject oriented: Ability to define a data warehouse by subject matter.
Integrated: All records or data from different sources must be in a consistent format. If all inconsistencies among records are resolved records, they can be called integrated.
Nonvolatile: Data should not change after their entry.
Time-Variant: Containing data from different time periods to identify trends (Inmon, 2005).

Ralph Kimball, the father of business intelligence, preferred a simpler 1definition with the focus on the function of the data warehouse. He defined it as a copy of transaction data specifically structured for query and analysis (Franks, 2013).

3.2.2 Data Mart

The term data mart is very often mentioned in the data warehousing field. Data marts serve the same role as data warehouse. However, they are intentionally limited in in scope and they usually serve one particular department or line of business. The advantage of data mart is that it can be created faster due to the limited scope. The disadvantage is that data marts make inconsistency problems because definitions and calculations need to be the same across data marts. There are two types of data marts, independent data marts and dependent data marts. The independent data marts are fed directly from source systems whereas dependent are fed from another data warehouse layers (Lane and Potineni, 2014). The data warehouse layers will be discussed later.

3.2.3 Operational Data Store (ODS)

Operational data stores are meant to support daily operations. The ODS data are cleansed and validated, but they store limited amount of historical data: it may be data for a day for example. The purpose of the ODS is to give the data warehouse the access to the most current data that has not been loaded into the data warehouse so far. Also they might be used as a source of data as for the data warehouse loading data in almost real-time manner (Lane and Potineni, 2014).

3.2.4 Operational vs. Analytical systems

It is important to realize that there is a difference between operational and analytical systems. The operational systems, sometimes called transaction processing systems, are simply where you put the data in while analytical systems (for example DW) are where you get the data out. The operational systems deal with transactions such as orders, new customers and complaints. They are designed to process transactions quickly and almost always deals with one transaction record at the time. Also, they conduct the same operational tasks over and over. Given this focus, they usually do not maintain history but rather update data to reflect the most current state. In contrast, the analytical systems watch

the functioning of transaction systems through evaluating their performance. They basically worry about transactional systems and provide information about them. They almost never deal with just one record at the time as they are optimized for high performance queries searching through hundreds or hundreds of thousands of transactions to deliver the desired answers. Another difference is that analytical systems typically store historical values to assess the organization's performance over time. As a result, those systems are designed in a different manner. Considering the differences, the analytical systems, including data warehouse, cannot be defined in a simplified manner as copies of operational systems on a different platforms (Kimball, 2013).

3.2.5 Data Warehouse Goals

Goals of data warehousing can be derived from needs and issues of companies related to storage, access, and analysis of their data. Organizations usually struggle to:

- Access data
- Identify the relevant data for decision making
- Compare data due to different formats
- See data in a broad view
- Use information to support more fact-based decisions.

Indeed, there are more issues that organizations have to face. However, these are recurring and frequent issues based on which the general goals of DW systems have been set (Ponniah, 2001).

The general requirements for a DW system are:

Accessibility: The DW system has to make information more accessible and understandable. It means not only to developer, but also to the business user who need to find the information simple and in a fast manner.

Consistency: The data usually come from more than one source so it has to be assured that they are cleansed, quality assured, and released once they are fit for usage. Otherwise, there can be many problems due to ambiguity.

Adaptability: User needs, business conditions, data, and technology are all subject to change so a DW must be adaptable. It also means that it should be possible to add new content without any disruption of existing data.

Presentable in a timely way: As the DW systems are used in an increasing manner for decision making, it is important that both DW teams and user have realistic expectations.

Security: As it has been said before, information is a valuable asset of any organization. For some, it is even the most important part of their business meaning that the DW system needs to be able to keep that information secure.

Trustworthy: A DW system must have a right to be the decision support system since the most important outputs are decisions made based on analytical evidence provide by the DW system.

Accepted: No matter how great the DW system is, it failed the acceptance test unless it is accepted by the business community (Kimball, 2013)

3.2.6 Data Warehouse Architecture

Common data warehouse architectures are based on layered approaches which will be analyzed in the following sections. Data warehouse developers may choose between those architectures depending on specifics of organization's situation (Linstedt and Olschimke, 2016). There is a number of architectures that can be found in organizations. However, the architectures to be analyzed are two-layer architecture by Ralph Kimball, three-layer architecture by Bill Inmon, and Data Vault architecture by Dan Linstedt because these people are of great influence in data warehousing. The architecture is introduced with emphasis on the main differences between them. The more detailed description of each of them, such as extensions and extra functionality, is not possible due to the limited scope of this thesis.

All of the architectures to be described are successfully implemented in the companies all over the world. All the mentioned architectures have enterprise focus. The enterprise focus means that they need to provide analytical support across the entire business or organization. That means that they need to address all the requirements within both divisional and corporate areas. Every architectural approach has a single integrated repository of atomic data. In the Corporate Information Factory, this repository is called the enterprise data warehouse. In the dimensional data warehouse, this repository is called

the dimensional data warehouse. The integrated nature of the central repository is consistent with an enterprise focus. It brings together various vantage points on common entities, such as customer or product. Likewise, its atomic focus addresses enterprise objectives. Data is not collected at the level of detail required by a particular group or subject area. Instead, it is collected at the lowest level of detail available, allowing it to satisfy any analytic requirement (Adamson, 2010). The main difference is how it is designed and used. Kimball advocates the use of dimensional design for the integrated repository of data (data warehouse layer) while Inmon advocates the integrated repository in the third-normal form. Lindstedt advocates rather Inmon's approach with some modifications as it will be seen later. Speaking of the subject-area level (data-marts), Inmon and Lindstedt promote separate physical data-marts whereas Kimball allows a logical construct. In other words, data marts can be just a subset of the integrated repository (Adamson, 2010).

3.2.6.1 Two-Layer Architecture

The two-layer architecture has been introduced by Ralph Kimball. This architecture, which is depicted in the figure below, consists of four layers from which only two are part of the data warehouse system itself - staging layer and data warehouse layer.

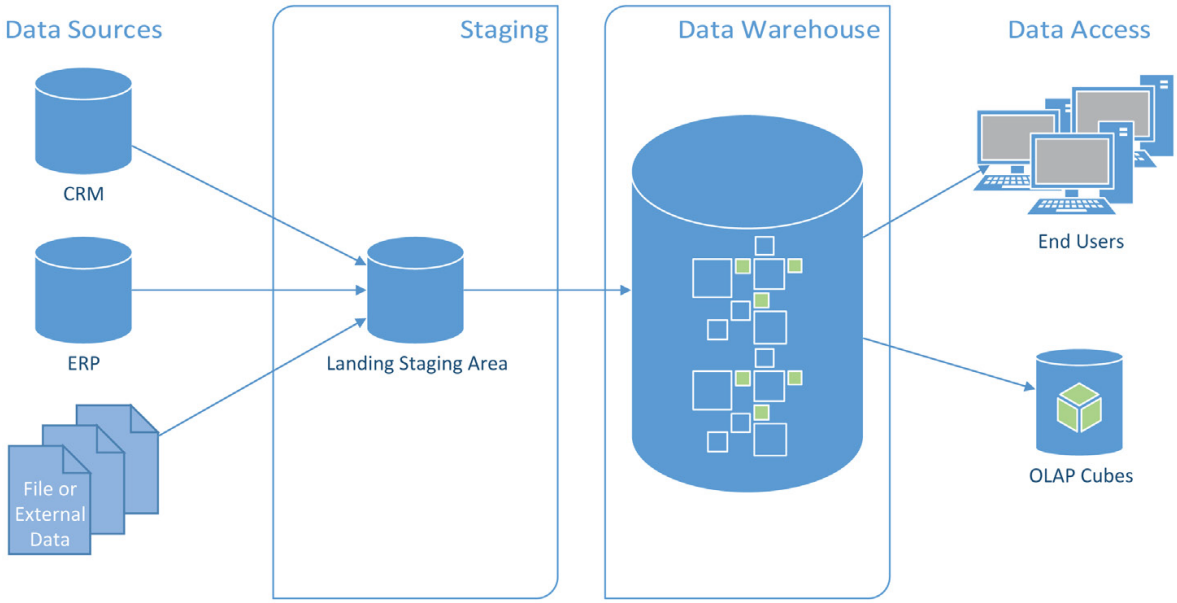


Figure 1: Two-Layer Architecture **Source: (Linstedt and Olschimke 2016)**

Data Sources

Data sources, also called operational source systems, record and capture business transactions. They can be thought of as internal or external data systems in which a user has a very limited or no control over the content and the data format. The main requirements for such systems are processing performance and availability. Operational queries against source systems are narrow, one-record-at-a-time queries that are restricted in their demands on the operational system. They are definitely not queried in the same broad and unexpected manner as DW systems are queried. Speaking of historical data, they usually store little or none. They are designed as applications without any commitment to sharing common data with other operational systems and without any thought given to the possible future integration (Kimball, 2013).

Although it is very often believed that data warehouse creation involves mere copying data from operational systems, nothing could be further from the truth due to different architecture of both systems. Data warehouse and operational systems have very different requirements in terms of workload, data modifications, schema design, typical operations, and historical data handling. One major difference is that data warehouses are partially denormalized to optimize querying and analytical performance unlike operational source systems which are designed in a normalized manner to optimize the update-insert-delete performance (Lane and Potineni, 2014). Since data from operational source systems usually comes in different formats they need to be unified, cleansed and integrated before they can be loaded in the data warehouse.

Staging Area Layer

Staging area is the place into which raw data from source systems are loaded in order to be transformed and consequently loaded into the data warehouse. The main purpose of staging area is to reduce the number of operations on the source systems and the time to extract the data from it. The tables in the staging area are modelled after the tables in the source systems (Linstedt and Olschimke, 2016). Staging area is a subset of the extract, transformation, and load (ETL) system. ETL system consists of staging area, instantiated data structure and set of processes. In the figure 1, the ETL system is represented as the staging area and the arrows leading to and from it. The ETL system can be defined as everything between the operational data sources and the data warehouse area, also called

the presentation layer (Kimball, 2013). Extraction is the first step of the ETL system. Extracting means reading and understanding the source data and copying them into the staging area for further manipulation. After the data is extracted to the staging area it is ready for potential transformations, such as the data cleansing (misspelling corrections, resolving domain conflicts, dealing with missing elements, and parsing into standard formats), combining data from multiple sources, and data deduplication. The final step of the ETL process is loading into the presentation layer (Kimball, 2013). The ETL process is an essential part of the data warehouse construction since there is not no point bringing data over from operational data sources into the data warehouse layer without integrating it. If the data arrives into the data warehouse layer unintegrated it cannot be used for relevant analysis which is one of the main points of the entire data warehousing solution. (Inmon, 2005) The ETL process is a very complex and difficult issue to deal with in every environment, however, it is mandatory that the data flowing to the data warehouse are cleansed, integrated, and de-duplicated (Ponniah, 2001).

Data Warehouse Layer

Data warehouse layer, also called the presentation area, is the area where the data is organized, stored, and made available for direct querying by its users (Kimball, 2013). Once the data has been extracted and transformed in the staging area, it is supposed to load the data into the data warehouse. In this two-layer architecture, this data warehouse is modelled in a dimensional way and is made up of data marts representing the business processes that are bound by together by conformed dimensions. The dimensional model is easy to query and by users and analytical tools, such as OLAP front-ends or engines. The main advantage of the two-layer approach is the simplicity of building a dimensional model from the source data as compared to other architectures (Linstedt and Olschimke, 2016). The dimensional modelling and data marts will be discussed in detail later in this thesis.

Data Access Layer

Data access applications, also called BI applications, are the final layer of Kimball's two-layer architecture. It refers to applications that business users can take advantage of to leverage the data warehouse area for analytic decision-making. By definition, all BI

applications query the data warehouse area for improved decision-making. A BI application can be a simple ad-hoc query tool as well as a sophisticated data mining tool (Kimball, 2013).

3.2.6.2 Three-Layer Architecture

Three-layer architecture is another commonly used architecture. It is based on three layers and it has been introduced by Bill Inmon. The architecture is depicted in the figure 2 below.

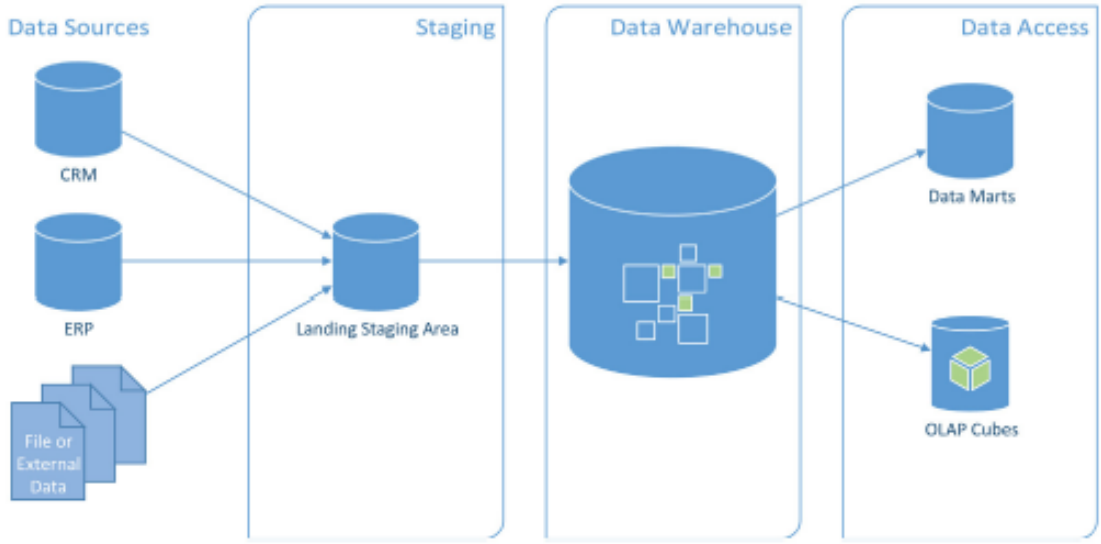


Figure 2: Three-Layer Architercture **Source: (Linstedt and Olschimke 2016)**

It is also known as Corporate Information Factory (CIF) advocated by Inmon and also other people in data warehousing industry (Kimball, 2013). The staging area in the three-layer architecture follows the one of the two-layer architecture. However, the main difference is in the data warehouse layer that holds raw data in a third-normal form (level of normalization). It integrates all data of the enterprise so it acts similarly to a large operational database. On the top of this normalized view there are subject-oriented data marts modelled in a dimensional manner. Users can access and analyze these data-marts similar to the two-layer architecture. However, it is much less complicated to build new data-marts since the data in the data warehouse layer are already cleanse and integrated so it is not necessary to go through the cleansing process. When it comes to disadvantages of this approach, it is much more complex and time-consuming to build the entire data

warehouse layer in a normalized form, including the dependent data marts. Another problem is that changes in data model may become a burden if too many data marts depend on the normalized layer (Linstedt and Olschimke, 2016).

3.2.6.3 Data Vault Architecture

Data Vault architecture is a three-layer architecture introduced by Dan Linstedt and is depicted in the figure below.

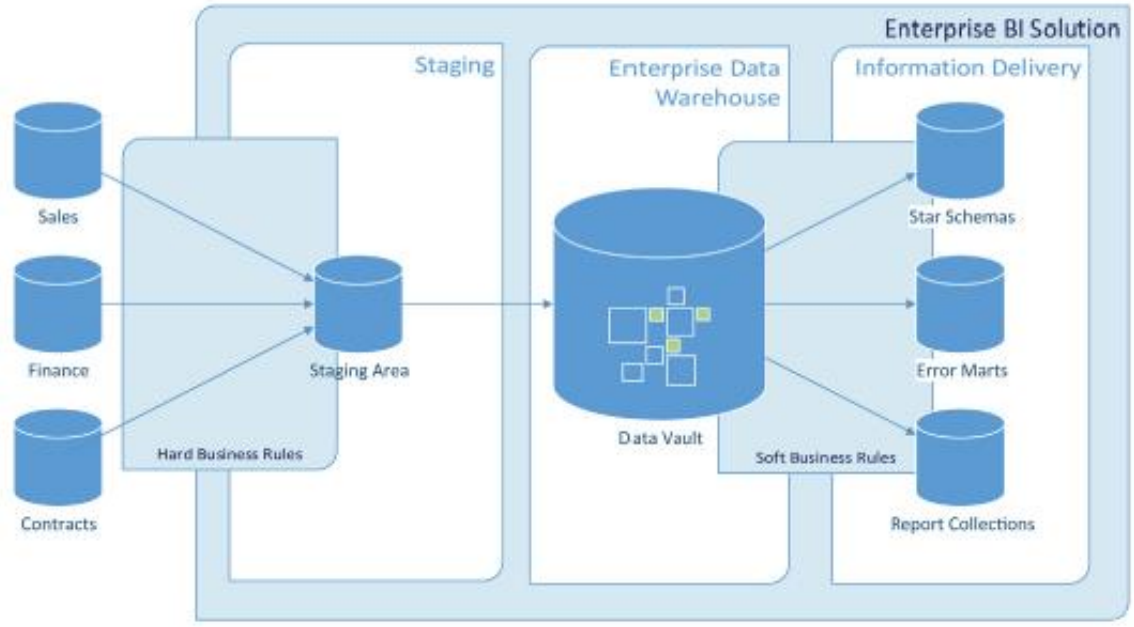


Figure 3: Data-Vault Architecture **Source: (Linstedt and Olschimke 2016)**

Business Rules

This architecture distinguishes hard and soft business rules. Generally speaking, business rules are constraints to incoming data to fit the requirements of the business. The difference between hard and soft data is that hard business rules never change the meaning of incoming data, only the way they are stored. In other words, they are only concerned with enforcement of data types. On the contrary, soft business rules change the meaning of the incoming data, for instance by modifying the level of detail or interpretation. Typical example is aggregation in categories or consolidation of data from multiple sources. They also determine how data is transformed to be meet the business rules. This architecture promotes enforcing hard business rules when loading a staging area unlike the two- and three-layer architectures that also apply soft business rules early in the loading process due

to the fact that the ETL data flows need to meet the business requirements. This early implementation improves the common application of the rules and the data quality. However, there are problems when the business rules changes because of dependencies in higher layers of the data warehouse (Linstedt and Olschimke, 2016).

Staging Area

Unlike the previously mentioned architectures, this one does not contain historical data. Instead, only the batch to be loaded into the data warehouse layer is present in the staging area. However, there might be multiple batches if some batches fail to load. The key purpose of having no history in the staging area is that source tables may change which would become a very complex over time and the idea of this architecture is to move complex business rules towards the end-user in order to be adaptable to changes. This staging consists tables that duplicate the structure of the sources systems. It keeps the same tables, columns, and primary keys. However, objects ensuring the referential integrity are not duplicated - indexes and foreign keys. On the top of that, all columns are nullable so that the data warehouse can load the raw data from the source systems including the bad data. The only business rules applied are the hard business rules. In addition to the columns in the tables from the source systems, each table in the staging area includes a sequence number, timestamp, record source, and hash key computations for all the business keys and computations (Linstedt and Olschimke, 2016). These fields are metadata which are required for loading the data into the next layer. The sequence number is a generated numeric value that serves as a data identifier (Sequence Numbers, 2016). The timestamp is automatically generated date and time of when the data arrives in the data warehouse. It basically identifies the arrival of data (Timestamp, 2016). The record source indicates the original data source of data and the hash key is used for identification purposes (Linstedt and Olschimke, 2016).

Data Warehouse Layer

The data warehouse layer, which is called Data Vault 2.0 by Linstedt, stores all historical and time-variant data. It holds raw data not modified by any business rules other than hard business rules so data is stored in the same level of detail (granularity) as provided from the source system. Nonvolatile means that every change in the source system is tracked by

the data warehouse layer. Unlike the two- and three-layer architectures, the data warehouse layer is not directly accessed by end users. Typically, the end users only access the data marts that are easier to navigate for them (Linstedt and Olschimke, 2016). The Data Vault can be also defined as a detail oriented, historical tracking and uniquely linked set of normalized tables that support one or more functional areas of business. It is a hybrid approach that aims to combine the best features of two- and three-layers architectures taking advantage of the 3rd normal form (3NF) and dimensional modelling (Data Vault Basics, 2016).

Data Access Layer

Linstedt uses the term information layer and information marts instead of data layer and data marts as he emphasizes the delivery of the information. As it has been said previously, the end-users access only this layer to get the information. The layer is subject-oriented while the data vault is function-oriented. It often follows the principles of dimensional modelling and forms the basis for both dimensional modelling and online analytical processing (OLAP). However, the information marts in this layer can also be modelling in the third normal form to fit the preferences of the end users. In this layer, there are two special marts that are different to the rest - the Error and Meta marts. They are central repositories of errors and metadata, respectively. End users, such as administrators, use them to analyze a number of problems in the data warehouse. Unlike information marts, they cannot be rebuilt from the Data Vault or any other source (Linstedt and Olschimke, 2016).

3.2.7 Kimball vs. Inmon Architectures

These two architectures are very often put against each other so it is vital to compare them (Adamson 2010). Inmon’s architecture requires longer start-up time, higher initial costs and it is more time-consuming to build. However, once it has been built, it is easier to maintain and further development costs are lower. Kimball’s architecture takes lesser time to develop and initial costs are lower, but each subsequent phase costs the same and it lacks the complex enterprise focus. Overall, both architectures have its advantages and disadvantages, however, Inmon’s is more oriented on large enterprises (ComputerWeekly, 2012).

3.3 Database Design

As it has been defined in the previous chapters a data warehouse is a relational database that can include a number of components (Lane and Potineni, 2014). The relational database makes a significant part of the entire data warehouse environment so it is important to understand its design and modelling techniques. The database design incorporates requirements analysis, logical design, physical design, and database implementation and monitoring.

3.3.1 Requirements Analysis

The requirements for the database design are assessed by interviewing both producers and users of data and using the information to produce formal requirements specification. This formal specification includes the data for required for processing, the natural data relationships, and the software platform for the database implementation (Teorey, 2011).

3.3.2 Logical Design

After the requirements analysis, the collected requirements need to be translated into a database system. To do this, the logical and physical design need to be created. The logical design focuses on the logical relationships among objects whereas physical design tries to find the most effective way of storing and retrieving objects as well as handling them from a transportation and backup/recovery perspective. Therefore, the logical design is more conceptual and abstract than physical (Lane and Potineni, 2014). One way to model organization's logical information requirements is entity-relationship (ER) modeling. Another may be through the Unified Modeling Language (UML). These techniques deal with identifying the things of importance, the properties of these things and how they are related to each other. They are purely logical and apply to both transactional systems and data warehousing systems, however, they are also applicable for the various physical design techniques (Lane and Potineni, 2014).

3.3.2.1 Entity-Relationship (ER) Model

The entity-relationship (ER) approach was first presented in 1976 by Peter Chen. Chen's approach uses rectangles to specify entities, which are somewhat analogous to records. It also uses diamond-shaped objects to represent the various types of relationships, which are differentiated by numbers or letters placed on the lines connecting the diamonds to the rectangles (Teorey, 2011). ER model can be defined as a database modelling tool used by database designers who must communicate with end-users about their data requirements. In general, there are two forms of ER models, the simple form and the complex form (Teorey, 2011). The simple form is preferred by both database designers and end-users since it is easy to use and understand. The basic ER model contains three classes of objects which are entities, relationships and attributes. Entities represent the principal data objects about which information is to be collected (Lightstone et al., 2007). They usually denote individuals, places, things, event or information of interest. A particular occurrence of an entity is defined as an entity instance. Employees, projects, departments, skills or locations can be all considered examples of entities. The entity construct is a rectangle inside which there is its name. Another object of a ER model is a relationship which represents a real-world association between among one or more entities. The relationship construct is a diamond connecting associated entities (Teorey, 2011). Relationships are described in terms of degree (recursive binary, binary and ternary), connectivity (one-to-one, one-to-many, many-to-many), and existence (optional or mandatory). The most common meaning related to the relationship is indicated by the connectivity, see the figure 4.

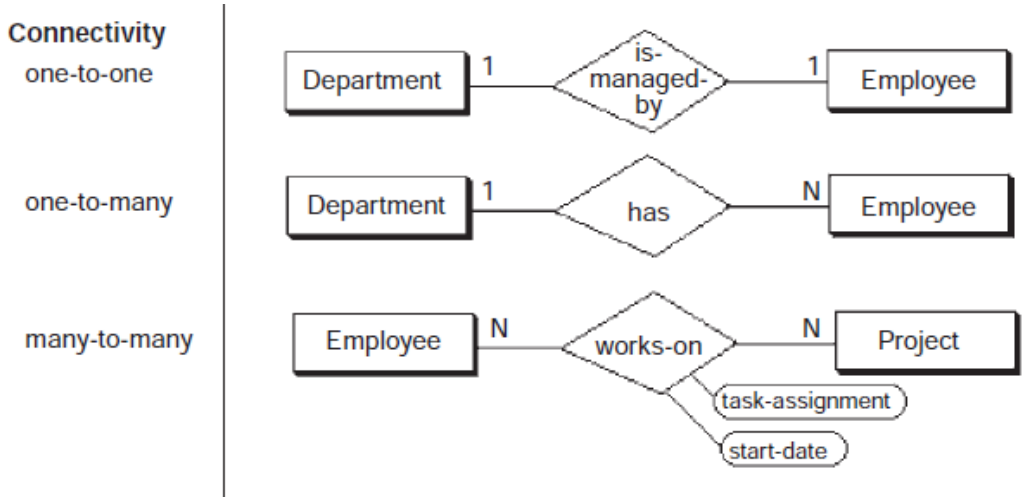
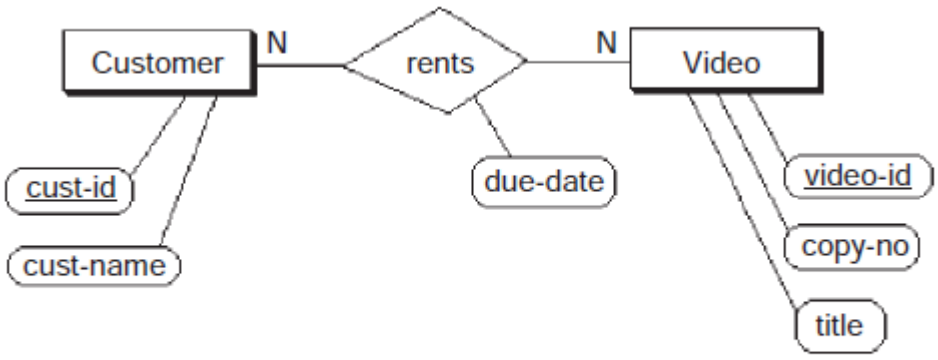


Figure 4: Relationship Connectivity **Source: (Teorey 2016)**

The connectivity of a relationship constraints the number of the entity occurrences in the relationship. Values for connectivity are either “one” or “many”. In the figure …, one is represented as 1 and many as N. In the one-to-one case, one department is managed by one employee where the minimum and maximum value for both department and employee is one. In the one-to-many case, one department is associated with many employees where the maximum connectivity for the employee is the unknown maximum N and the minimum connectivity is exactly one. On the department side, both maximum and minimum is one. In the many-to-many case, both sides have the minimum connectivity one and the maximum connectivity unknown N. The actual count of elements associated with the connectivity is called the cardinality of the relationship connectivity; it is used much less frequently than the connectivity constraint because the actual values are usually variable across instances of relationships.

The last object of a simple ER model is an attribute. Attributes are essentially characteristics of entities that provide descriptive details about them. A particular occurrence of an attribute is an attribute value. There are two types of attributes, identifiers and descriptors. Identifiers, also called keys, are used to uniquely identify an instance of an entity. On the other hand, descriptors are used to specify a non-unique characteristics of instances. The attribute construct is an ellipse with the attribute name inside. To better understand a simple ER model, the figure 5 representing it is provided below.



Source: (Teorey 2011)

Figure 5: A Simple ER Model

In this very simple example of the ER model, customers and videos are entities with the relationship “rents” showing a many-to-many relationship between them. Both entities

have a number of attributes describing them. The “cust-id” uniquely identifies the “customer” entity and “video-id” uniquely identifies the entity “video”. The rest of attributes are non-unique characteristics.

There are also more complex ER models that helps database designer to capture more details about the model without long explanations. However, more details make it more difficult for the end user to understand the model so the basic model is recommended as the communication tool for the conceptual database design verification. The complex ER models are very rarely used and have no generally accepted form yet (Teorey, 2011).

3.3.2.2 UML

The Unified Modelling Language is another frequently used tool for conceptual database design. It was introduced in 1997 by Grady Booch and James Rumbaugh and has become a standard graphical language for specifying and documenting large-scale software systems. The UML (now UML-2) has a great deal of similarity with the ER model. The goal of conceptual schema design, where the ER and UML approaches are most useful, is to capture real-world data requirements in a simple and meaningful way that is understandable by both the database designer and the end user (Teorey, 2011).

3.3.2.3 Schema

A schema is a collection of database objects, including tables, views, indexes, and synonyms. These objects can be organized in the schema models in many different ways. An important part of the schema design is to decide whether to use the third normal form, star, or snowflake schema. Most data warehouses take advantage of dimensional modelling techniques.

3.3.2.4 Third Normal Form Schemas

Third Normal Form (3NF) design is a design approach aiming to minimize data redundancy and avoid anomalies in data insertion, updates and deletion. The third normal form is reached through the process of normalization which will be analyzed in the following part. The third normal form design is very often used and widely accepted as optimal for transactional processing systems that need to maximize performance and accuracy when inserting, updating and deleting data (Star Schema, 2016). However, for

analytic systems such as data warehouses, the primary interaction style is querying which may be very complicated since normalized databases use very many tables that require complex queries. Many tables in a query mean more potential data access paths that slow down query performance.

3.3.2.5 Normalization

Normalization is a data design process that has a high level goal of keeping each fact in just one place to avoid data redundancy and insert, update, and delete anomalies (Haan et al., 2011). Normalization can also be defined as the process of decomposing large, inefficiently structured tables into smaller, more efficiently structured tables without losing any data in the process. Normalization supports the proposition that a well-defined database contains no duplicate data and keeps redundant data to an absolute minimum. This, in turn, guarantees data integrity and ensures that the information retrieved from the database will be accurate and reliable (Hernandez, 2003). There are multiple levels of normalization, however, for the purpose of this thesis, it makes sense to describe only those needed for reaching the third normal form - first-normal form, second-normal form, and third-normal form. A relational database table is often described as normalized if it meets third-normal form (Date, 2004). Speaking of the normal forms, each subsequent normal form depends on normalization steps taken in the previous normal form. For example, to normalize a database using the second normal form, the database must first be in the first normal form. Before the normalization process, each table must have a primary key (unique identifier) and a table cannot contain repeating groups of data. Each table needs to be in this order to do the normalization process effectively (Hernandez, 2003)

First-Normal Form

The purpose of this Normal Form is to ensure that a table does not contain any multipart or multivalued fields and that each field holds only a single value for any given record.

Second-Normal Form

The second-second normal form makes sure that each non-key field in the table is functionally dependent upon the primary key and that the table does not contain calculated fields.

Third-Normal Form

The third-normal form makes sure that:

- Each field value is independently updateable; changing the value for one field in a given record does not adversely affect the value of any other field in that record.
- Each field identifies a specific characteristic of the table's subject.
- Each non-key field in the table is functionally dependent upon the entire primary key
- The table describes one and only one subject (Hernandez, 2003).

3.3.2.6 Denormalization

Denormalization is an approach to optimize data retrieval in which specific instances of redundant data are added back after the data structure has been normalized. A denormalized database should not be confused with a database that has never been normalized (Denormalization 2016). Denormalization can also be defined as the consolidation of database tables to increase performance in data retrieval (query), despite the potential loss of data integrity (Teorey, 2011).

3.3.2.7 Star Schema

The term star schema is another way of referring to dimensional modeling approach to defining a data model. The dimensional design for a relational database is called a star schema (Adamson, 2010). The star schema gets its name from its appearance because it resembles a star where the fact table is in the center and dimensions around the fact table. Dimensional modeling is based on creating multiple stars, each based on a particular business process. The mechanics of dimensional design is very simple. It supports analysis of a business process by modelling how it is measured. The goal of dimensional modeling is structural simplicity and high performance data retrieval (Lane and Potineni, 2014). It is basically optimized for analytic systems. As it has been analyzed, the third-normal form minimizes data redundancy and the risk of insert, update and delete issues. In contrast, the dimensional modeling accepts data redundancy, denormalization, for the sake of data retrieval performance and understanding (Adamson, 2010). The modern approach to data warehousing does not put the third-normal form and the star schema against each other but

display them as complementary to each other. In a dimensional design, measurements are called facts and context descriptors are called dimensions (Lane and Potineni, 2014).

3.3.2.7.1 Dimension Tables

In a star schema, a dimension table contains columns representing dimensions. Sometimes, dimension table is shortened to dimension. Generally, you can tell from the context whether dimension is a column of a table (Adamson, 2011). Dimension tables provide context for the fact table as depicted in the figure 6 below.

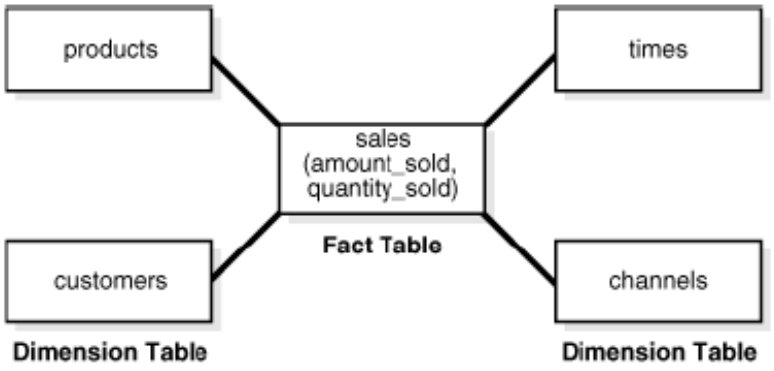


Figure 6: Star Schema **Source: (Lane and Potineni 2014)**

For example, in the figure above dimension tables product, customers, times, and channels provide detail for the fact tables. They serve as look-ups or reference tables. Also, they can be used for data filtering. Dimension data typically stores rows for the lowest level of detail and rows for aggregated dimension values. These natural rollups or aggregations within a dimension table are called hierarchies and add great value for analyses (Lane and Potineni, 2014). Each dimension table is given a surrogate key which is a unique identifier created exclusively for the data warehouse. Surrogate keys have no meaning and are usually integers. The surrogate key is the primary key of the dimension table. While dimension tables have far fewer rows than fact tables, they can be quite wide, with dozens of columns (Lane and Potineni, 2014). There are also special cases of dimension tables, degenerate dimensions and junk dimensions. Degenerate dimensions are dimension columns stored in a fact table which cannot be assigned to any dimension because they are unclear (Adamson, 2011). Junk dimensions collect miscellaneous attributes that do not share common affinity (Lane and Potineni, 2014).

3.3.3 Physical Design

The physical design of a database starts with the schema definition of logical records produced by the logical phase (Lightstone, 2007). During the physical design process, there is conversion of logical design structures into physical ones. Physical design is mainly by query performance and database maintenance aspects (Teorey, 2011). It looks for the most efficient way of storing and retrieving data as well as handling them from a transportation perspective (Lane and Potineni, 2014). There is also a simplified definition of physical design saying that it is essentially creating of the database with SQL statements. Among others, the following structures are important parts of physical design: indexes, partitioning, clustering, views, integrity constraints (Lane and Potineni, 2014).

3.3.3.1 Partitioning

Partitioning is a method of reducing the workload on any hardware component. The result of partitioning is a balanced workload across the system and preventing bottlenecks (Lightstone, 2007). It can also be defined as division tables, indexes, and index-organized tables into smaller pieces where every piece is called a partition. This division gives database administrators significant flexibility. The fundamental for partitioning strategies for partitioning are list, range, and hash partitioning (Lane and Potineni, 2014).

3.3.3.2 Multidimensional Clustering (MDC)

MDC is a technique by which data can be clustered by dimensions such as product type or location. This clustering technique takes advantage of the anticipated of expected workloads of on this data (Lightstone, 2007).

3.3.3.3 Indexes

Index is a data structure defined on columns in a database table to significantly speed up data retrieval operations at the cost of additional storage. They are used to find data without having to search every row in a database table every time a database table is accessed. Indexe can be created on one or more table columns (Index (IDX), 2016). The common indexing techniques are b-tree indexing and bitmap indexing (Lane and Potineni, 2014).

3.3.3.4 Views

A view is database object generated from a query and stored as a permanent object. Although the definition says it is permanent, the data contained in the view dynamically changes depending on the point of access (View, 2016). Views are used, for example, to provide additional security by restricting access to the base tables, to hide data complexity or present different perspective of the base table (Ashdown and Kyte, 2015).

3.3.3.5 Integrity Constraints

Integrity constraints are used to enforce business rules defined at the column or object level restricting values in the database. Integrity constraints are used to, for example, enforce business and data loading rules to assure integrity. An example of a constraint is NOT NULL disallowing inserts or updates of rows containing a null in a particular column (Ashdown and Kyte, 2015).

3.3.4 Database Implementation and Monitoring

Database implementation and monitoring is the last step of the database development. Once the previous parts have been completed the database can be created through a number of SQL statements. The database is created using Data Definition Language (DDL) commands. Then, it can be queried and updated using Data Manipulation Commands (DML) commands as well as to set up indexes, integrity constraints and other objects (Lightstone, 2007).

3.4 Data Warehouse Development Methodology

In this chapter the methodology of building the data warehouse is analyzed. The discipline studying the process that people use to develop information system is called the system development life cycle (SDLC) or the system development methodology. There are two main variants - waterfall and iterative. The waterfall methodology is also known as the sequential methodology. The iterative methodology is sometimes mentioned as the spiral or incremental methodology. The waterfall methodology is one of the oldest methodologies and is widely used by a number of organizations when building their systems, including data warehousing systems (Rainardi, 2008). Iterative methodology is

very different from the waterfall methodology which will be seen in the following analysis of both. Besides the differences, advantages and disadvantages of every methodology will be presented.

3.4.1 Waterfall Methodology

In the waterfall methodology, there are steps that need to be accomplished one way after the other, in a particular sequence. There are many variations of the waterfall methodology, however, the usual steps are feasibility study, requirements, architecture, design, development, testing, deployment, and operation. The figure 7 below, enriched with project management and infrastructure steps, describes the steps in order (Rainardi, 2008).

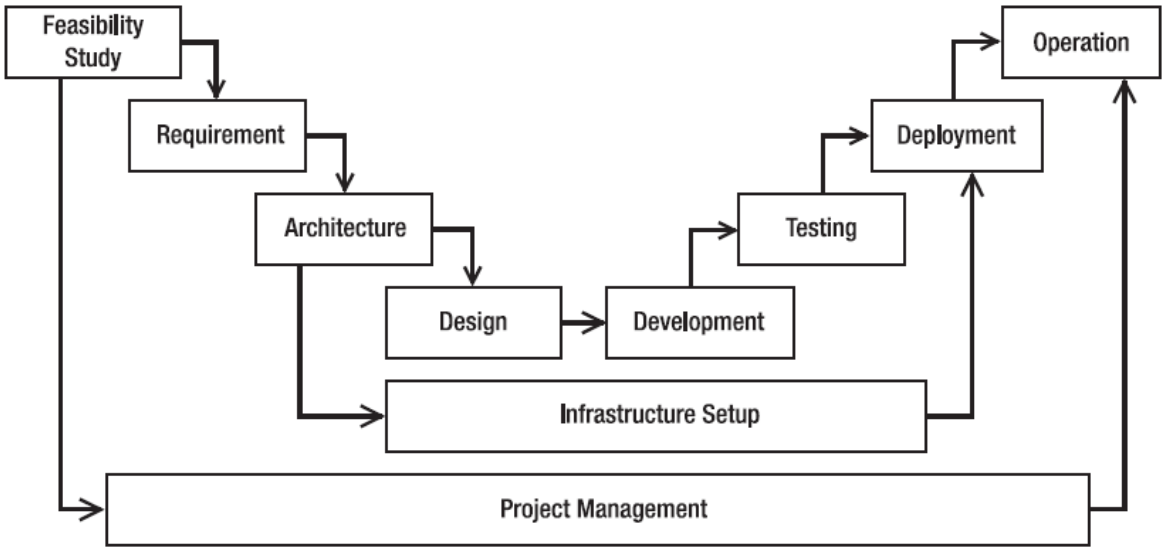


Figure 7: Waterfall Methodology

Source: (Rainardi 2008)

There can be variations of the step names such as initial investigation, system design, implementation, operation, rollout, or go live. Some of these variations are synonyms, others turn bigger steps into smaller ones (CMS, 2005). To explain the meaning of the mentioned steps:

3.4.1.1 Feasibility Study

Feasibility study represents gathering the requirements at a high level. There is an analysis of the source systems to determine whether it is possible to get the needed data. Also,

assessment of the data quality is done. Based on the results, a proposal, also called a business case, is written. This proposal contains benefits, costs, duration of the project, and business justifications. Optional parts are a requirements summary, project organization, and project plan. This step requires a lot of experience since the people involved in the study needs to know the consequent steps in advance in order to produce a decent proposal (Rainardi, 2008).

3.4.1.2 Requirements

Requirements are gathered through talking to users to understand the details of business data and processes. Besides, analysts need to identify issues and also list the nonfunctional requirements such as performance and security.

3.4.1.3 Architecture

High-level architectures were described in one of the previous chapters. Basically, it is necessary to decide which system architecture will be used including other technical specifications such as storage solutions or BI tools

3.4.1.4 Design & Development

These steps involve design and development of the data warehouse solution. They involve data modelling and other techniques analyzed in the previous chapter to construct the data warehouse components.

3.4.1.5 Testing

The data warehouse components need to be tested to reveal potentially weak points in the whole project. For the waterfall methodology, this is the most dangerous moment of the entire development process since it is for the first time that all the components are assembled together and the entire architecture runs as a system. If there are too many issues during the testing part, it might be very hard to fix them in time because the deployment is approaching very fast and having problems in the production environment cannot be risked. These problems can be partly mitigated by employing a prototyping step earlier in the process.

3.4.1.6 Deployment

Once all the problems have been fixed, the system is ready for the deployment. It means that all DW components are put into the production. It is made sure that every part works as supposed, guides are provided, and users are trained. Then, the project is handed over and the project can be closed.

3.4.1.7 Operation

The users continue to use the DW and its applications. The support is provided to solve errors and help current and new users. Users have very often some requests concerning adding new functionality to the data warehouse system.

3.4.1.8 Infrastructure setup

One of the most challenging tasks is to build a production environment where the data warehouse is run. The tasks consist of creating system architecture, setting technical design, installing and configuring software, connecting networks, testing the infrastructure, producing documentation, and passing it over to the operational team. This step is very often underestimated and results in the delays in the project deployment. This step can be considered as a project itself due to its scale and importance. At the very least it should be considered as a subproject (Rainardi, 2008).

3.4.1.9 Project Management

A project is a temporary activity with a defined beginning and end in time, and therefore scope and resources. It is unique in the sense that it is not a routine operation, but a specific set of operations designed to accomplish a singular goal. Project team often includes members that do not work together. Since the project needs to be delivered on time it needs to be managed which is exactly what project management is about. Project management is about application of skills, knowledge, tools, and techniques employed in the project to meet the project requirements (PMI, 2016).

3.4.2 Iterative Methodology

The key principle of the iterative methodology is to deliver growing portions of the complete project to discover issues early rather than deal with them towards the end of the project. In the waterfall methodology, there is only one period of testing before deployment. On the other hand, the iterative approach is based on building the solution in iterations so the potential issues can be discovered and fixed earlier (CMS, 2008). The figure below depicts the situation when a solution is built in three iterations. Each cycle of the iterative methodology consists of design,

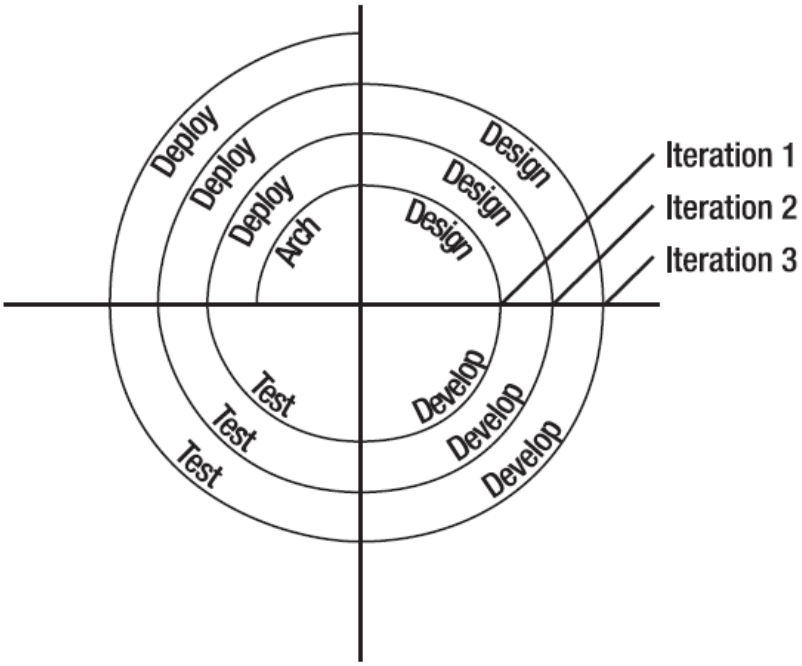


Figure 8: Iterative Methodology **Source: (Rainardi 2008)**

development, testing, and deployment. The architecture is completed only in the beginning. Feasibility study and requirements are included in the architecture in this case. The main reason to employ this methodology is to minimize the risk of the first release during the testing step which is very close to deployment (Rainardi, 2008).

3.4.3 Comparison of Methodologies

There are many variables and specifics in every data warehousing projects that affect selection of the right methodology. One has to consider a number of factors including

system type, project team composition, team members experience, project manager experience, solution scalability, and clarity of objectives. These factors together with specifics of every methodology should be taken into consideration (CMS, 2008).

3.4.3.1 Advantages and Disadvantages of Waterfall Methodology

In terms of advantages, it is easier to plan, design, and measure the development of the project because the project is divided into exact phases, strict time schedules and deadlines. Another advantage is that the customer's presence is not required a number of times for testing as in the case of iterative approach. The only exceptions is the presence for review, approvals and status meetings. Finally, the solution can be designed in a more complete and careful way because there is a focus on the final solution not on the pieces where parts are added after every iteration (Lotz, 2013).

As far as the disadvantages are concerned, this methodology is inflexible, slow and costly due to significant structure and tight controls. The entire project depends on the requirements gathered in the beginning of the project yet users are not always able to clearly define their requirements. This may result in the situation that customers do not like the solution in the end and the potential changes are hard and costly. The fact that there is a limited room for testing increases the likelihood of inconsistencies and missing components in the production version (CMS, 2008).

3.4.3.2 Advantages and Disadvantages of Iterative Methodology

The main advantages of the iterative methodology are reduced project risks, improved participation of users, resolving unclear objectives, and ability to launch incomplete but functional product (CMS, 2008). It addresses the inability of users to specify their requirements by providing them a system for experimental purposes. The methodology is very useful for resolving unclear objectives by developing and validating user requirements after each iteration (Lotz, 2013).

One the other hand, there are also disadvantages. High degree of participation may slow down the project. Also, some users may not be interested for increased participation (Lotz, 2013). A number of iterations contributes to the fact that requirements can change very often which may be reflected on the quality of the product. Another problem is that

designers may substitute prototyping for sound design and underestimate up-front analysis which leads to poor design in both cases (CMS, 2008).

3.4.4 Trends in Data Warehousing

Organizations all over the world have been centrally storing their data in data warehouses for more than 30 years. In the 1990s, there was a big trend of data warehouses that were of terabyte in size. Nowadays, state of the art warehouses are much bigger storing petabytes of data. Data warehouses are not just bigger though. They are faster, support new data types, provides wider range of business functions and are also more accessible (Foley, 2014). Customers have also been digitally empowered so they have bigger power to decide about organizations fate. As a result, businesses try to understand their customers to give them what they want. This desire for customers understanding has given rise to the concept of business intelligence (BI), the use of data mining, big data, data analytics and other tools to analyze data in a faster and efficient way to gain the competitive edge. However, these tools and other technologies related to the data processing need to continuously evolve to keep pace with data trends. This is also driving the future of data warehousing (van Loon, 2016). It is not possible to encompass all trends due to scope reasons. However, some of the trends are:

3.4.4.1 More Capable Data Warehouses

A growing number of devices generating data in various formats forces IT teams to add new capabilities to data warehouses so they can handle the new data type, more data, and also process them faster (Foley, 2014).

3.4.4.2 Data Warehouses and Hadoop

Hadoop, which is a framework that allow for the distributed processing of large data sets across clusters of computers using simple programming models (Apache Hadoop, 2016) is increasingly used as a complementary solution to data warehouses thanks to its ability to process large data sets (Foley, 2014). There is also pursuit to leverage Hadoop's unparalleled power to run analytics directly in cluster (Marvin, 2015).

3.4.4.3 Cloud-Based Data Warehousing

Cloud computing which can be defined as internet-based computing, where different services, such as servers, storage and applications, are delivered to an organization's computers through the Internet (Cloud Computing, 2016) shows the growing adoption in the data warehousing because of the growing demand for online databases and other related services (Henschen, 2015).

3.4.4.4 In-Memory Architecture

The emergence of in-memory database architecture brings increased performance to data warehousing. This architecture is based on processing large data sets in random access memory (RAM), accelerating data processing (Foley, 2014).

4 Analytical Part

This part of the thesis is based on the real-world data warehouse implementation in a bank in Georgia. The exact name of the bank cannot be provided due to a non-disclosure agreement so the bank will be referred to as the "Georgian Bank". As indicated in the beginning of this thesis, the main purpose of this part is to analyze the implementation process of a data warehouse in a banking environment and to find the optimum solution for the bank which means, in this case, to find the best database management system (DBMS) using the multi-criteria decision analysis. To reach all the objectives that this thesis had set, this part was divided into three sections - Introduction of the Company, Data Warehouse Implementation and Selection of the Optimum Database Management System.

4.1 Introduction of the Company and Case

4.1.1 The Georgian Bank

The Georgian Bank was originally a state-owned bank which was privatized in the middle of the 1990s. Since then, it has become one of the five largest banks in Georgia with almost 7% market share as of 2016. In 2015, it served approximately one and half a million individuals and more than 60 legal entities. Speaking of the capital structure, it is a joint stock company listed on the Georgian Stock Exchange. To briefly introduce the bank in terms of the financial performance, the Georgian Bank has been profitable in terms of net income for more than six years.

4.1.1.1 Strategy

The Georgian Bank's main objective is to become the market leader in terms of number of customers and financial performance by increasing its market share and focusing on the maximization of shareholder profit. To be more specific about the strategy, the key aspects of the strategy are as follows:

- Modernization of its branches: The Georgian Bank aims to gain a competitive edge by modernization of its branches and finding attractive locations for them.

- Upgrade of alternative and traditional services channels: The Georgian Bank looks to innovate the traditional service channels as well as to take advantage of emerging distribution and service channels.
- Diversification of revenues and profitable growth: The Georgian Bank tends to maintain and increase its revenues through diversification of its profits and targeting new socio-demographic strata.
- Financial structure optimization: The Georgian Banks aims to broaden its funding to decrease reliance on the funding from the public sector institutions.

4.1.1.2 Management and Ownership Structure

As far as the management form is concerned, the Bank of Georgia follows the model of supervisory and management board. In the supervisory board, there are four board members and the executive chairman. The management board consist of six members responsible for their divisions and one member leading them. Speaking of the ownership structure, the majority (about 60%) of shares is held by a private financial group and the rest by minority shareholders.

4.1.2 Business Case

Due to its long-term strategy seeking growth in the previously mentioned areas of business and interest in efficiency maximization, the Georgian Bank decided to analyze its functioning to receive recommendations on its functioning. The analysis highlighted a number of issues related mainly to poor data management practices and weak Enterprise Architecture that were preventing the Georgian Bank from reaching its goals and more efficient functioning. To be more specific about the results of the analysis, a very thorough analysis indicated many issues from which only the ones relevant to the implementation of the data warehouse will be presented:

4.1.2.1 Inefficient Reporting

There was a significant number of analysts preparing reports of very limited relevance due to missing data management practices. It was analyzed that there was no business dictionary providing directions on how to calculate specific attributes across the company. A product catalogue allowing easy client segmentation and transaction classification was

also missing. Master Data Management, or methods enabling the bank to link all of its critical data into a master file, that provides a common point of reference was also missing. Data quality checks assuring the optimum quality of data were also insufficient so the data used for reporting could not be considered of great significance.

4.1.2.2 IT Teams Segregation

The analysis also indicated that IT teams of the Georgian Bank worked in isolation and without any common rules on how to work with data which resulted in poor data quality and significant reporting issues. To give an example, there were data about customers in more than fifteen databases. Every database stored data about customers in various formats and with the different level of detail resulting in serious reporting difficulties.

4.1.2.3 Dependence on certain IT specialists

The analysis also indicated that the Georgian Bank was too dependent on certain leading IT specialists whose departure could seriously impact the functioning of the bank. Their departure would cause serious difficulties to the bank since these specialists were very often the only ones who knew how certain information systems work. At the same time, it was analyzed that a significant number of workers underperform.

4.1.2.4 Frailty IT Architecture and Practices

Another key issue was that the IT architecture was frailty and not built for the growing complexity. Considering the growth and the strategy of the bank and also its need for scalability, this frailty architecture could have fatal impacts on the bank's functioning in the mid- and long-term horizon. As far as the IT practices are concerned, the bank lacked architecture standards, proper documentation, data government guidelines, and strong IT governance processes.

Based on the analyzed issues, a solution was provided. A significant part of the solution was involved a data warehouse implementation that could, if implemented effectively, provide the Georgian Bank with the following improvements:

- **Unification** of multiple data sources into a single database that would be used to present data
- **Data integration** would enable combining data residing in different source systems and providing its users with unified view
- **Consistent information** on the key organizational data
- **Maintenance of data history** necessary for efficient analysis and decision-making
- **Data quality** improvements
- **Advanced analysis** and predictive modeling including segmentation, predictive analysis, propensity analysis, and revenue forecasting
- **Self-documenting code** helping prevent human sluttery

Following the results of the analysis, the Georgian Bank decided to give the data warehouse project the green light. The entire data warehouse project implementation is closely analyzed in the following part.

4.2 Data Warehouse Implementation

After the data warehouse project was given the green light its implementation could begin. In terms of the system development life-cycle (SDLC), it was decided that the project would be managed using iterative methodology for a number of reasons. First, this methodology is generally accepted as modern and the most efficient in the data warehousing field and within the development team. Second, the project took part in Georgia which was an unknown business environment at the time so it was extremely necessary to produce results in a fast manner and to react to the customer quickly because of his potentially changing requirements. This would be very complicated to achieve applying the waterfall methodology. The iterative methodology applied during the implementation consisted of six parts as indicated in the figure …below:

- Analysis & Requirements
- Architecture
- Design & Development
- Testing
- Deployment

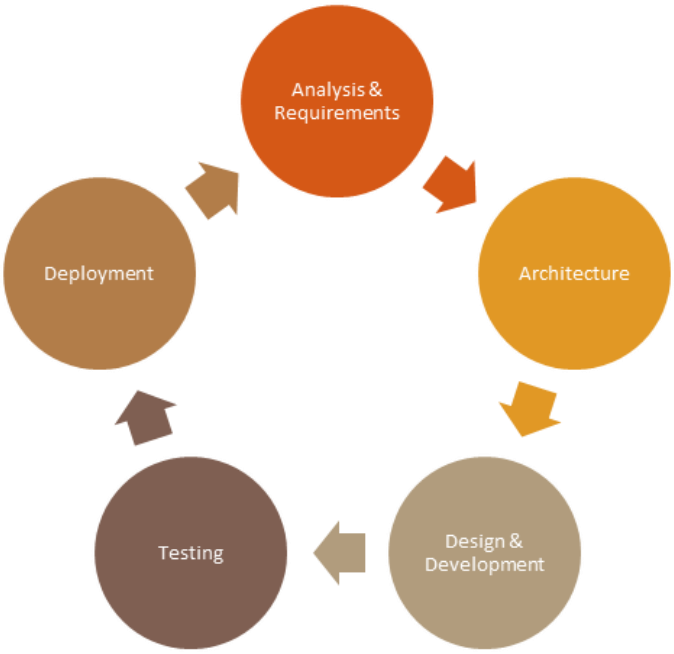


Figure 9: System Development Life Cycle **Source: Own Input**

To clarify the SDLC, the data warehouse project was delivered in three iterations. These iterations will be introduced later in this thesis. The first iteration followed all the steps that are depicted in the figure 9 above. However, the second and the third iterations were mostly concerned with enrichment of the first iteration having so the Analysis & Requirements and the Architecture phases was omitted from the life cycle because there was no point going through them again. Therefore, the second and the third iterations followed slightly modified SDLC that is depicted in the figure 10 below.

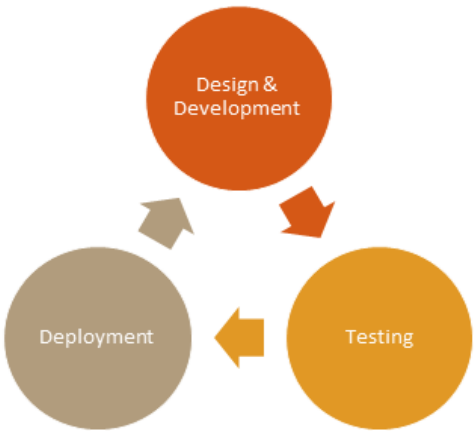


Figure 10: Modified SDCM **Source: Own Input**

4.2.1 Analysis & Requirements

This first step of SDLC was extremely important in the beginning of the project as further steps of the data warehouse implementation were either directly or indirectly dependent on this step. During this phase, business requirements and technical requirements were gathered as well as the complex definition of the project was provided. All key business areas, operations, and data sources were identified by interviewing individuals, observation of workers and analysis of business documents and reports. Based on this analysis, the following key objectives were identified:

- Make reporting and ad-hoc analysis faster and more efficient across all departments in the bank
- Provide advanced reporting solution for central reporting and risk management

In order to reach the desired goals, it was needed to identify the key business data and to build the data marts for risk and central reporting purposes. The key data was concerned with:

- Accounts
- Applications
- Transactions
- Clients
- Collaterals
- Products

As far as data marts are concerned, the ones with the highest priority are those for central reporting and risk management, however, the warehouse should be built with regard to the eventual design of other data marts. After the identification of key business data and the data marts, it was necessary to conduct a feasibility study assess the viability of these components.

4.2.1.1 Key Data Sources Evaluation

All key data and data marts in terms of issues, requirements, and definition of successful completion of that activity. Due to the limited scope of this thesis, one data mart and one representative of key business data will be analyzed as examples of this study.

Application Data Analysis

Application data is either stored in an insufficient form or not at all which makes the analytics of the approval process very complicated. Therefore, it is very hard to analyze the number of approved, rejected, new, cancelled, modified, and completed applications. Besides, identification of employees working with applications is complicated.

Users Requirements

Resolve the current issues and make it possible to interconnect applications with clients and products data. Also, integration of the data sources is needed for more efficient reporting.

Completion Requirements

In order to satisfy users' requirements, the application data needs to be gathered and stored in an integrated repository with the following attributes:

- Client ID
- Loan / Contract ID
- Date and Time of Application
- Application Reason
- Application Status
- Application Scoring
- Application Decision
- Application Approval/Rejection Reason
- Application Result
- Loan / Contract Purpose
- Application Channel
- Business User Involvement

- Application Variant
- Application-Client Relationship
- Application-Contract Relationship
- Application-Product Relationship

Successful Completion

The requirements are satisfied only if all data is gathered, stored and provided for the consequent analysis.

Central Reporting Data Mart Analysis

As analyzed, report generation is time-consuming and inefficient. The data used for central reporting are stored in many databases in different forms which makes the reporting inconsistent and decision making distorted.

Users Requirements

Users required to have a repository of data for central reporting purposes that will make their work faster and more efficient. The managers of the company requested more precise reports so they could make better strategic decisions.

Completion Requirements

In order to build a central reporting data mart, it is needed to gather and integrate all the data about clients, accounts, applications, products and collaterals. This data needs to be aggregated into a data mart that will efficiently present data to business users. The key aggregations are:

- Products
- Balances
- Transaction
- Channels
- Loan Types
- Delinquencies
- Clients

Successful Completion

The successful completion of the central reporting data mart involves gathering and integration of the mentioned data and delivery of the data mart that will provide snapshots of key facts and dimensions.

4.2.1.2 Results Evaluation

Such analysis was done for every key data source and data mart to be built to make sure that the project is technically feasible and has benefits for all the parties involved. The thorough analysis indicated that the project was technically feasible because the required data is accessible and the time needed for the completion of the project was not too long to be beneficial. Also, the project was identified that the cost-benefit value is positive for both parties involved - customer and developer. After both parties agreed on the final price and scope of the project further steps of the data warehouse implementation could begin. To achieve all the projected goals, it was suggested that the core data warehouse project would be accompanied by another project running in parallel. The core data warehouse project was concerned with the technical solution which is the analytical data store (ADS) where all the data will be stored in multiple layers. Speaking of the parallel project, it would be focused on setting up enterprise architecture and business optimization that would assure the sustainability of the new data warehouse and amplify its contribution in the long-term. This thesis is focused on the core data warehouse project, however, it is also important to mention this parallel project to see how complex issue the data warehouse implementation is and to take into consideration its specifics. This parallel project was supposed to introduce new enterprise architecture standards and practices that would be used during the approval of new business requests, design of new solution and products, and producing documentation to take full advantage of the new solution. Besides, it was meant to optimize the current processes and prepare workers for the new data warehouse in terms of routines and procedures.

4.2.1.3 Project Timeline

The data warehouse project was planned to last for twelve months during which the data warehouse, also referred to as the analytical data store (ADS) during the project, would be delivered in three iterations.

First Iteration

The first phase was projected to last for six months. During this phase, the following goals were set to be achieved:

- Initial analysis and data sources identification and analysis
- Selection of the optimum database management software and tools for data warehouse design and maintenance in the Georgian Bank.
- Overall architecture of the data warehouse and its layers.
- Design and development of all data warehouse layers - staging area (L0), central repository area (L1), and data mart area (L2)
- Testing and Deployment

Second Iteration

This iteration was projected to last three months. This iteration was expected to accomplish the following goals:

- Enrichment of the layers L0, L1, and L2 built during the first iteration. It was supposed that business users would take part in testing and deployment to specify what data is missing.
- Design and development of new data marts related to collections and customer relationship management (CRM)
- Data quality improvements
- BI Reporting
- Testing and Deployment

Third Phase

The third phase was very similar to the second one in terms of objectives and length. It was expected that it would last for three months. During these three months the following objectives were supposed to be accomplished:

- Further enrichment of the layers L0, L1, and L2 to the final form
- Design and development of a regulatory data mart
- Data quality improvements
- BI Reporting

4.2.1.4 Project Team

An appropriate project team needed to be formed in order to successfully conduct the implementation of data warehouse. The proposed project team consisted of two sub-teams that were expected to work together - delivery and customer team.

Delivery Team

The delivery team was supposed to be responsible for the development of the data warehouse. It was expected to be composed of a delivery manager, solution architect, developers, analysts and programmers. To be more specific about team members and their roles, brief description is provided.

Delivery Manager: There was one delivery manager responsible for organization, planning and supervision of other team members. Also, he was supposed to deal with any obstacles that were preventing other team members from delivering their work in time.

Solution Architect: The solution architect was the one with overall responsibility for the data warehouse life cycle which means analysis, design, development, testing, and deployment.

Senior Developers: There were two senior developers subordinated to the solution architect. Their responsibility was to conduct the database development in the most

efficient way as well as continuously gather requirements from the customer team during the development phase.

Junior Developers: There were two junior developers that were subordinated to the senior developers. Their responsibility was to develop database layers as well as do testing to make sure that the solution works in a proper way.

Business Analyst: There were two business analysts in charge of analyzing banking processes and products relevant to data warehouse development. Their main role was to bridge business issues and technology solutions.

Reporting Analyst: Initially, the reporting analyst was responsible for analyzing current reports and sharing his findings with developers so the developer solution is more relevant. Later on, his role shifts to generation of reports.

Programmers: The programmers were responsible for development and configuration of database development software.

Customer Team

Members of the client team represented the Georgian Bank. This team was composed of project leader, product owners and reporting specialist. The project leader's responsibility was to deliver the agreed amount of information to the supplier team and to manage his team. Then, there were product owners representing their departments - sales, collections, marketing etc. Their responsibility was to define business requirements and to test the developed data warehouse components. Finally, there were reporting specialists that were supposed to answer questions asked by delivery team members concerning source systems and reports.

4.2.2 Architecture

The architecture of the data warehouse reflected the size and requirements of the Georgian Bank as well as the overall objectives of the project. The figure 11 below show the

simplified architecture of the data-warehouse, also referred to as the analytical data store(ADS).

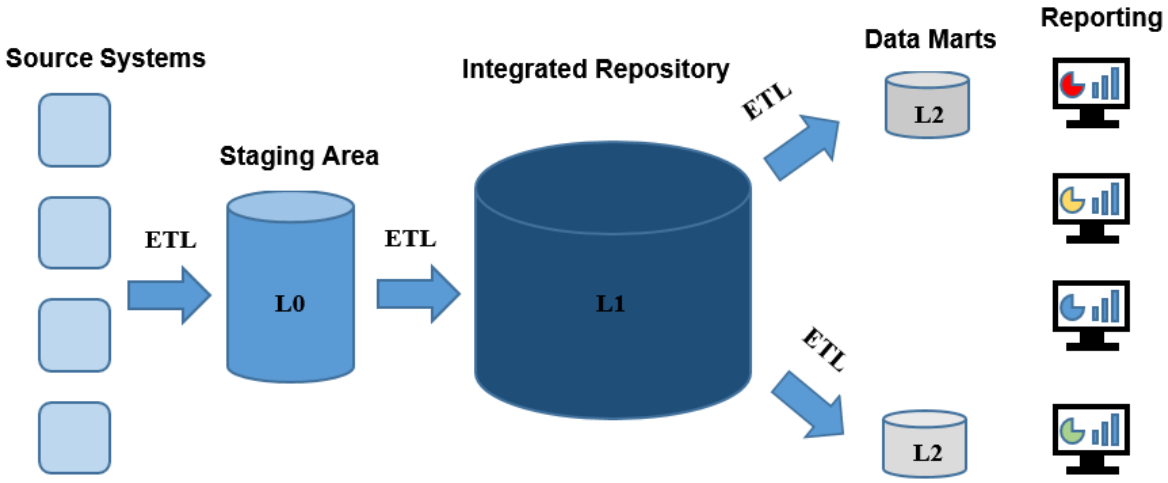


Figure 11: Data Warehouse Architecture **Source: Own Input**

As it can be seen from the picture, the data warehouse architecture consists of five components, also called layers:

- Source systems (SRC)
- Staging area (L0)
- Integrated repository (L1)
- Data marts (L2)
- ETL
- Reporting

4.2.2.1 Source Systems

This layer contains source data for the data warehouse layers. There is data coming from either internal (banking systems) or external systems (credit bureau) of the company. It was analyzed that there were five major databases from two different providers - two databases from Microsoft and three databases from Oracle. The data stored in these databases is stored in various formats and level of detail and it is practically impossible to analyze them due to lack of integration.

4.2.2.2 Staging Area (L0)

This layer is positioned between the source systems and the integrated repository. It serves as a storage of daily extracts from the banking internal and external source systems. This data is consolidated, transformed and prepared for the further load into the integrated repository which will be discussed in the next section. The data stored in this layer is inconsistent, has no history (only for the retention period), and has the same structure as the source systems. Therefore, the main purposes of having the source system data in this layer are to:

- Consolidate data from multiple internal and external source systems
- Implement transformations that cannot be done on the source system level
- Decrease the number of operations on the source systems
- Add metadata such as timestamps

4.2.2.3 Integrated Repository (L1)

As its name indicates, this repository integrates the data from the staging area. The data in this layer is supposed to be the central source of integrated, historized and audited data for the data marts. Also, the data can be directly queried in this layer if testing is needed otherwise these layers are supposed to be locked for end-users. The structure of this layer is fixed and is supposed to reflect the business processes and relations of the Georgian Bank. The data stored in this layer are:

- Integrated and consistent
- Cleansed
- Containing history
- Subject oriented

From the data modelling perspective, the data held in this layers are stored in a normalized manner. However, they are not stored purely in the third-normal form. The third normal

form would decrease the number of duplicate values, however, it would decrease performance and complicate querying due to the need to join more tables.

4.2.2.4 Data Marts (L2)

The data mart layer (L2) can be seen as a subset of the L1 layer that focused on certain business departments or processes. The data contained in the data marts are loaded through ETL process. From the perspective of data modeling, the data is modeled in a dimensional way which means that there is a number of star schemas and snowflake schemas with fact tables representing business processes and their dimensions representing the context. This layers accepts duplication and denormalization for the sake of simplicity and query performance. To sum up the feature of data in this layer, they are:

- Focused on the certain business departments or processes. For instance, central reporting and risk management after the first iteration
- Easy to read and query by end-users
- Optimized for frequently run queries

In the case of the Georgian Bank, the number of data marts grows with every iteration. The first iteration brings data marts focused on central reporting and risk management. The second delivers data marts dealing with CRM and collaterals. The third iteration introduces the data mart specialized on financial reporting.

4.2.2.5 Extract-Transfer-Load (ETL)

ETL is a complicated and time-consuming process. It takes care of the data movement between the data warehouse layers. In the case of the Georgian Bank, the data is transferred at the end of each day in batches from the source systems to the L2, through L0 and L1 layers. The entire process will be demonstrated in the next section.

4.2.2.6 Reporting

The reporting layers represents all the analytical applications used for the analysis of data. In the case of the Georgian Bank, such applications are Microsoft Excel and Microsoft Power BI. The source of data for these applications are data marts.

4.2.2.7 Tools used for Architecture and Design & Development

The implementation of data warehouse was a very complex process during which a significant number of tools needed to be used. These tools are listed and briefly described below:

DBMS - Oracle 12c

Oracle 12c was chosen as the database management systems for the Georgian Bank. It is the latest DBMS from Oracle. The process of selection of this DBMS is analyzed in the next chapter of this thesis.

PL/SQL Developer

PL/SQL Developer was used to develop and manage the Oracle 12 DBMS as it offers full functionality of running queries and scripts, managing database, a report interface, a data modeling solution and also a data migration platform.

SAP PowerDesigner

SAP PowerDesigner is a modeling tool that was used during the data warehouse implementation process. Data models of the data warehouse as well as the mappings, ETL code and packages were developed and generated by this tool. The figure 12 below demonstrates the graphical user interface.

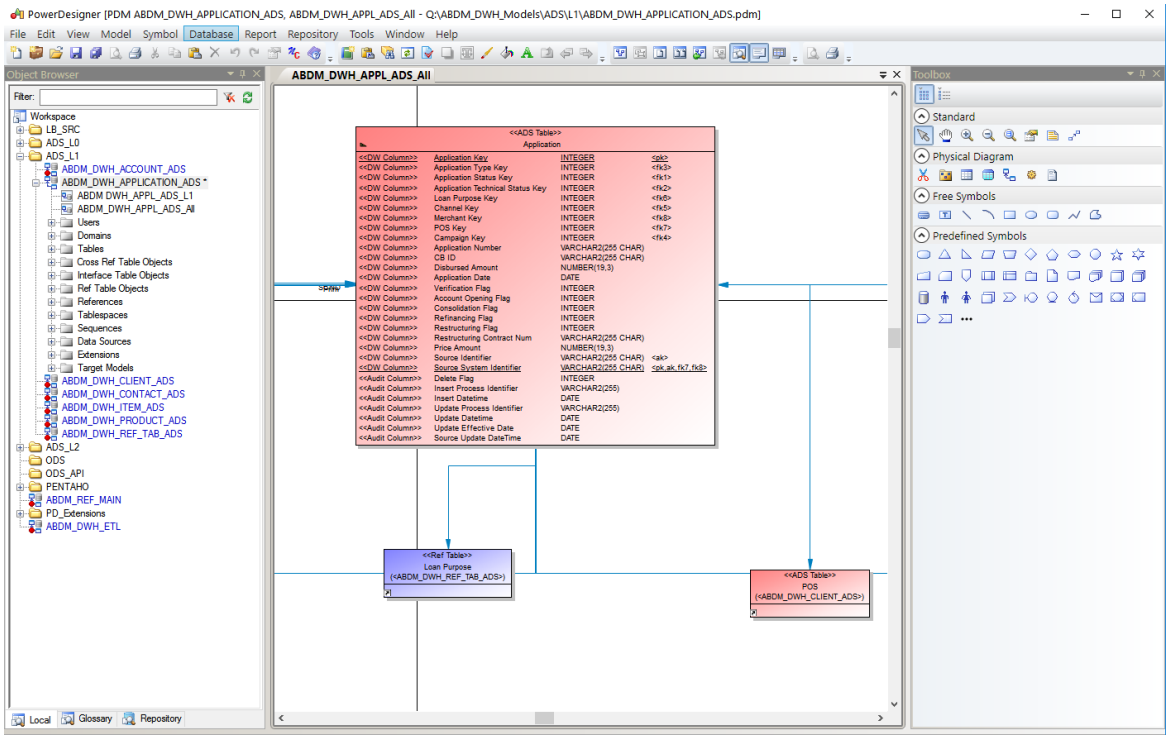


Figure 12: Power Designer User Interface **Source: Own Input**

ETL Tool - Pentaho Data Integration

There was a number of ETL tools to choose from. However, when the requirements were specified the range of options shrank to three possibilities - Oracle Data Integrator, Pentaho Data Integration and Clover ETL. The tool was required to handle workflow management and execution of ETL packages generated by SAP PowerDesigner. Finally, Pentaho Data Integration was chosen as the best option in this particular case.

BI Tool - Power BI

Power BI is a business analytics solution provided by Microsoft. This solution enabled end-users to create their own reports and dashboards without any need for technology staff or database specialists. As far as the version is concerned, Microsoft's cloud based analytics version was selected.

The further steps of the SCDM will only be concerned with the first iteration of the data warehouse project. The main reason is the limited scope of this thesis as well as the fact that the second and third iteration mostly deal with enrichment of the already existing components that are fully functional. Indeed, the analysis of the all three iteration would

provide more complex picture of the data warehouse implementation. However, the data warehouse is supposed to be fully functional after the first iteration so skipping on the second and third iterations would not have negative impact. To clarify the Design & Development process, the development stages did not wait for the preceding stage to be completed. At some point, there were four stages in a parallel development.

4.2.3 Design & Development

This part of the data warehouse implementation was concerned with the design and development of the data warehouse components introduced in the architectural part. The entire Design & Development process could be divided into seven stages:

- Reverse engineering of the Source Systems (SRC)
- Design & Development of the Staging Area (L0)
- Design & Development of the ETL process from Source Systems to L0
- Design and Development of the Integrated Repository (L1)
- Design & Development of the ETL process from L0 to L1
- Design & Development of the Data Marts (L2)
- Design & Development of the ETL process from L1 to L2

4.2.3.1 Reverse Engineering of the Source Systems (SRC)

The first step of the design and development process started with reverse engineering of the Georgian Bank's databases. It was done using the SAP PowerDesigner's function "reverse engineer" that allowed to connect to the Georgian Bank's live databases and generate physical data models (PDM) from them. In the beginning of the project, it was analyzed that the Georgian Bank operated on five databases - three using Oracle technology and two using Microsoft technology. As a result of the reverse engineering there were more than sixty data models engineered from the above mentioned live databases. These physical data models showed all tables structures and relationships between their tables. The figure 13 below depicts relationships between tables from one data model running on one of the Oracle databases

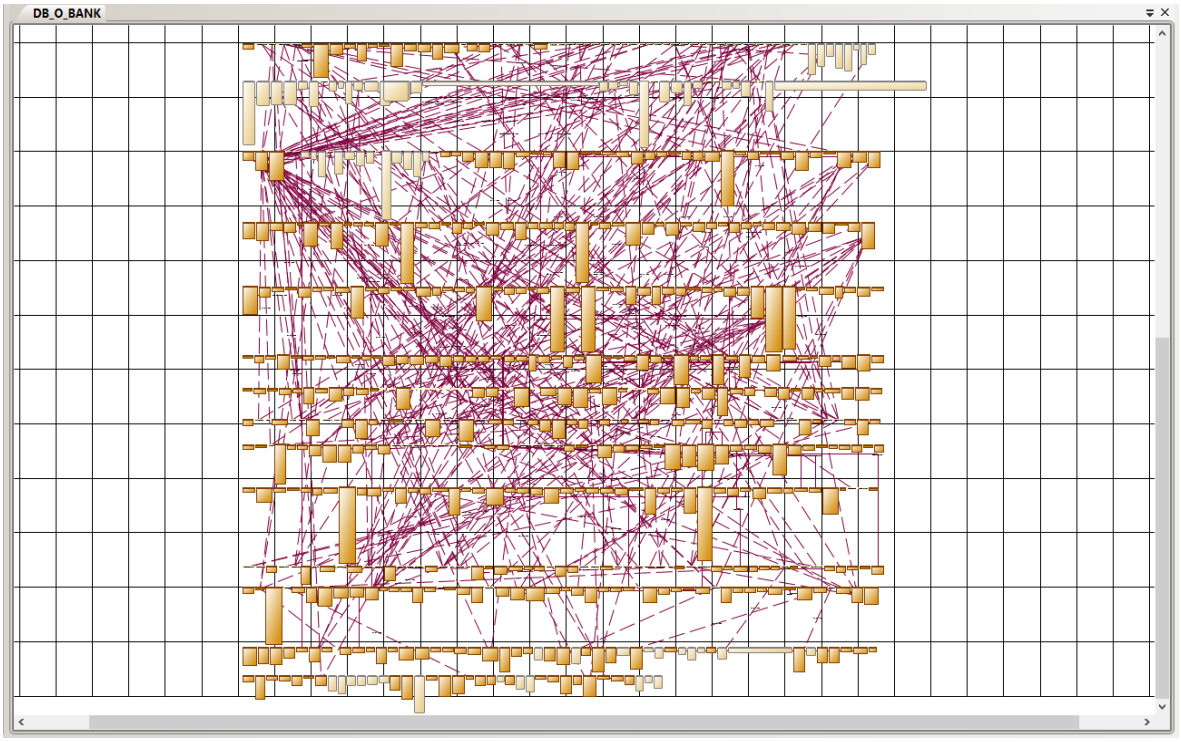


Figure 13: Source System Physical Model **Source: Own Input**

4.2.3.2 Design & Development of the Staging Area (L0)

After all the relevant source databases were reverse engineered and their physical data models loaded into the SAP PowerDesigner. The Design & Development of the Staging Area could begin. First, it was needed to identify what tables from the physical data models are needed to be moved into the Staging Area. The selection of these tables depended on the Georgian Bank's requirements and modeling needs. After the desired tables were selected, they were copied into the Staging Area (L0) in SAP PowerDesigner and subsequently modified. Speaking of the modifications, every table copied into the Staging Area undergo a number of modifications that would simplify the subsequent ETL process and further operations on these tables. To be more specific about the modification process, there was a set of procedures that needed to be done for every table copied into the L0. The first step was to set up a L0 data model for every source data model. It involved setting stereotypes, default and package owners, source data model name and abbreviation, and finally database link name. The figure 14 demonstrates such settings.

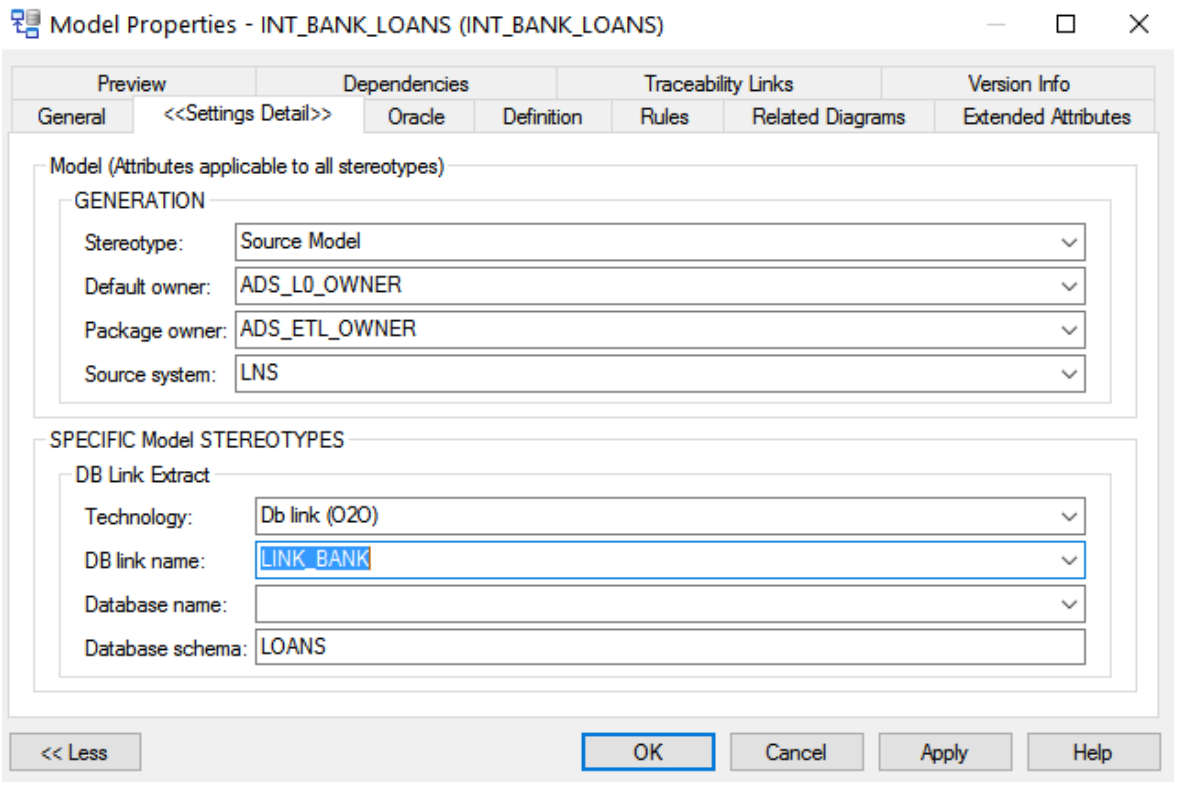


Figure 14: L0 Data Model Settings **Source: Own Input**

The second steps was to simply copy the source table from the source system layer and paste it into the L0 target model that had been previously set up. This was a very simple copy-paste operation in SAP PowerDesigner. After the desired tables were copied and pasted into the target L0 model, there was a set of modifications on the tables including:

- Adding Audit Columns
- Check and modification of table length
- Deleting legacy indexes, users, triggers, alternate keys
- Setting new stereotypes, users, permissions
- Adding table prefixes corresponding to the source systems models
- Setting scope and frequency of loads
- Setting the responsible analyst and person

Once the modifications were finished, the last step was to run checks to make sure that modifications are consistent.

4.2.3.3 Design & Development of the ETL process from SRC to L0

After all the tables were transformed in L0 layer, it was time to create them in the database using create scripts generated by SAP PowerDesigner. A create script is a SQL statement containing a SQL create table statement. Once these scripts created tables in a database it was needed to fill these tables with data. To do that packages for all tables in L0, containing SQL procedures, were created in SAP PowerDesigner and then validated in the database. Then, the packages could be run directly in the database using a simple script provided in the figure 15 below.

```
Begin
        NAME_OF_THE_PACKAGE.MAIN(P_CURRENT_DATE => to_date
        ('DATE','DATE_MASK"), P_JOB_ID => JOB_NUMBER);
End;
```

Figure 15: SQL Statement to call the procedure

Since there were more than five hundred packages it would be very complicated to run the packages one by one. That is the reason for the usage of the workflow management tools that ran all the packages in one job. The workflow management tool is used to launch the packages deployed in the database using a workflow diagram that was created in SAP PowerDesigner. The workflow schema is depicted in the figure 16. This simplified workflow schema consists of five components: start (green rectangle), common link (blue arrow), process (blue box), join (yellow rectangle), and end (red rectangle). To explain the mechanics, the start initiates the process so the packages 1,2,3,4 are run in parallel. If they are successfully finished, the and is completed and the process finishes. In reality, there are usually tens of packages that need to be launched in certain order to respect constraint-based loading.

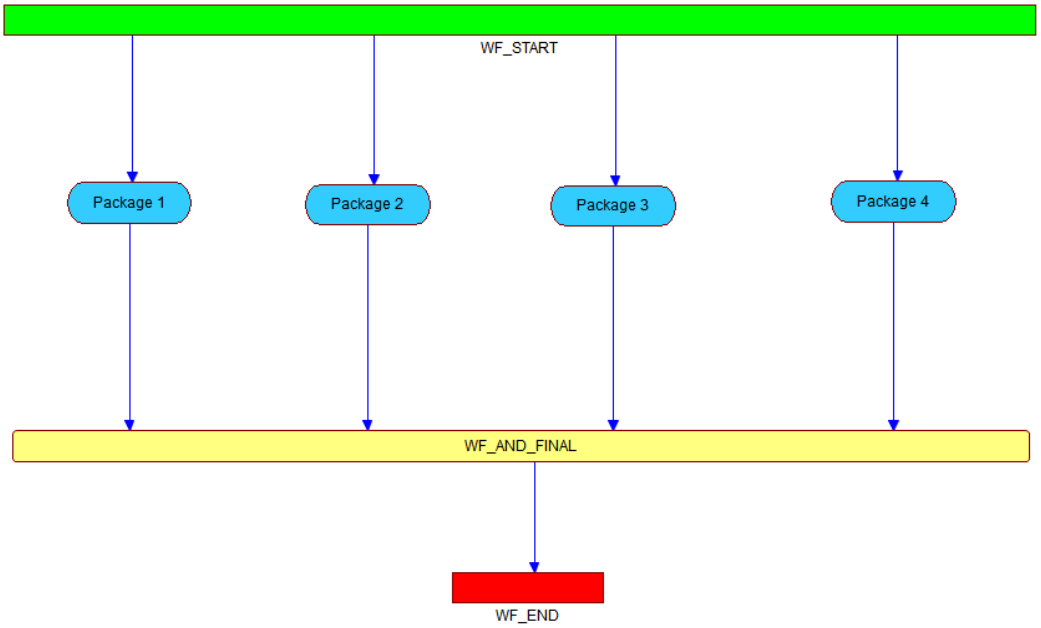


Figure 16: Sample Workflow **Source: Own Input**

4.2.3.4 Design & Development of the Integrated Repository (L1)

The next step was to design and develop the Integrated Repository (L1). As already mentioned in the architectural part, this layer is supposed to be the central source of integrated, historized and audited data. It was designed with regard to the reporting needs of the Georgian Bank. They are supposed to reflect the banking processes and functioning as well as to be the relevant source of data for the data marts in the L2 layer. There were six models created in a normalized manner, namely:

- **Account**: stores data about accounts and their balances, delinquencies, transactions, interest, provisions, and rates. There are also tables defining relationships with clients and products.
- **Application**: stores data about applications and their variants, scorings, approval reasons, approval decisions, approval statuses. Besides there are also tables that connect applications to products and clients.
- **Party**: stores data about clients and their relationships. In total, there are more than twenty tables in this data model concerned with clients.
- **Contact**: stores data relevant to the collection process and contact cases

- **Item**: items stores data about tangible objects including cars, collaterals, documents, financial instruments and goods. There are also tables concerned with relationships with clients and products,
- **Product**: stores data about banking products including loans, cards, insurance services and many others. There are also tables concerned with relationships with applications and clients.
- **Reference Tables**: This model contains data about reference tables that are used for looking up information.

As analyzed above, there were seven data models created in the L1 layers with more than one hundred tables which is a significantly lower amount of tables compared to the L0 layer which contains about six times more tables.

4.2.3.5 Design & Development of the ETL process from L0 to L1

This ETL process followed very similar procedures in terms of generating create scripts, ETL packages and running workflows. However, the entire process was much more time-demanding and complex. During the previous ETL process, one L0 table was loaded with data from one source table because L0 tables were copies of source system tables with certain transformations as described in one of the previous sections. This time though, it was needed to link a L1 table with different L0 tables that were corresponding to the meaning of the L1 table. This process is called data mapping in data warehousing. It was needed to do mappings for all the L1 tables and then generate ETL packages and run workflow. During the previous ETL process, the ETL packages were generic due to one-to-one mapping nature. The figure 17 depicts part of a mapping.

```
select
  CCK_ASL_INDI_INCOME.VALUE DOC_DATA ,
  C_DOCSTAT_ACT_KEY DOC_STAT_KEY ,
  C_DOCTP_CI_KEY DOC_TP_KEY ,
  ITEM.ITEM_KEY ITEM_KEY ,
  'DOCP.'||to_char(CCK_ASL_INDI_INCOME.ASL_PERSON_ID) SRC_ID ,
  'AAK1' SRC_SYS_ID /*C_SRC_SYS_BBK1_ID*/
from
  CCK_ASL_INDI_INCOME
  inner join ITEM on ITEM.SRC_ID='DOCP.'||to_char(CCK_ASL_INDI_INCOME.ASL_PERSON_ID) and ITEM.SRC_SYS_ID='AAK1'
where
  CCK_ASL_INDI_INCOME.INT_SNAP_DATE=C_CURRENT_DATE
```

Figure 17: SELECT of a mapping **Source: Own Input**

4.2.3.6 Design & Development of the Data Marts (L2)

There were two data marts designed and developed in this layer. The first data mart focused on the central reporting and the second on the risk management. Unlike the data models in L1, the data models in L2 were developed using dimensional modeling techniques. The data models in L2 aggregated key metrics relevant for the central reporting and the risk management reporting.

4.2.3.7 Design & Development of the ETL process from L1 to L2

The last ETL process that was developed to move the data from L1 to L2 followed the same methodology as the previous one. The only difference was that L2 layer was modeled in a dimensional way which required more frequent usage of SQL analytical function during mapping. However, these mappings between L1 and L2 models were not as complicated as the mappings between L0 and L1 since the data in L1 had been integrated so it was quite easy to navigate and find the needed data in L1 data models.

4.2.4 Testing

Testing was integral part of the design & development of all data warehouse components. This process of validating needed to be done to make sure that all the data models, ETL create scripts and packages meet the particular requirements. Also, the validity of data loaded to all the layers was tested to ensure data quality. There was a number of validation techniques, however, only the most essential ones are described below.

4.2.4.1 Data Models Testing

It was needed to run checks on the developed data models to make sure that they are modeled in the right way. A check, in this case, is a script validating data models tables and settings. To give an example, it makes sure that all table names fit in the desired length, indexes and partitions are set properly, and constraints are not missing.

4.2.4.2 ETL Packages Testing

ETL packages testing consisted of three steps. After the ETL package was generated by SAP PowerDesigner, it needed to be loaded into the database without any compilation

errors. Compilation errors indicate that there is something wrong with the code. The wrong code may refer may contain typing errors so it refers to non-existing tables or columns, for instance. Even after the package is created without any compilation errors it needs to be tested whether all the data constraints are complied with - NULL, UNIQUE, for example.

4.2.4.3 Data Validity

Once the data is loaded into the tables, there is a need to check for consistency and missing values. Unlike previously mentioned testing procedures, this sort of testing was very hard to automate and took a significant amount of time because a great deal of logic needed to be applied.

4.2.5 Deployment

Deployment was the last step of the system development life-cycle. After all the testing was done, the data warehouse components were put into the production to see how they work and what areas can be improved and enriched in the eventual upcoming iterations. Also, the end-users were trained to use the entire system and to maintain it on their own.

4.2.6 Summary of the SDLC

The first iteration of the data warehouse implementation delivered a functional data warehouse that increased both the quality of reports and the efficiency. However, the entire iteration got delayed by one month due to problems with data movement from the L0 to L1. These problems were mainly caused by the fact that there were many problems with source data recognition. It was very hard to navigate in source system models and to find the data with the desired meaning for the data models in L1. Also, there was much time spent on data validation.

4.3 Selection of the Optimum Database Management Software (DBMS)

The selection of the optimum DBMS was done using Multi-Criteria Decision Analysis (MCDA). In the beginning, there is a short introduction of this analysis. After, this analysis is applied on the particular case of the Georgian Bank.

4.3.1 Introduction of Multi-Criteria Decision Analysis (MCDA)

4.3.1.1 MCDA Definition

MCDA can be defined as a set of techniques with the goal of providing an overall ordering of options from the most preferred to the least preferred option. These options may differ in the extent to which they achieve several objectives, and no one option will be obviously best in achieving all objectives. In addition, some conflicts or trade-offs are usually evident amongst the objectives. MCDA is usually used when looking at complex problems that are characterized by a mixture of monetary and non-monetary problems. MCDA provides different ways of decomposing complex problems into smaller pieces, of analyzing and weighing them and their options and of reassembling them together to present a coherent overall picture. However, it is important to emphasize that the primary purpose of MCDA is to help in decisions-making, not to take decision (Multi-Criteria Analysis, 2009).

Types of Decision Problems

There are many MCDA methods which are suitable for a specific decision problems. Speaking of the decision problems, there are many problems that a decision-maker can face, however, they may be categorized into four main categories:

The Choice Problem: The goal is to select the single best option or to reduce a group of option to only one option.

The Sorting Problem: The goal is to regroup the options with the similar behavior or characteristics.

The Ranking Problem: The goal is to order options from best to worse by scores and comparisons.

The Description Problem: The goal is to describe options and their possible consequences.

It is also possible that a combination of the above mentioned decisions problems occur (Ishizaka and Nemery, 2013).

4.3.1.2 Selection of MCDA Method

There are many methods developed to deal with the problems that were introduced in the previous section, such as Analytic Network Process (ANP), Analytic Hierarchy Process (AHP), Potentially All Pairwise RanKings of all possible Alternatives (PAPRIKA), Promethee, Data Envelopment Analysis, and many more. Considering these and many other methods that have not been cited, it is very complicated to choose and difficult to justify the optimal method because none of the methods are perfect nor can they be applied to all type of problems. Each method has its advantages and disadvantages and there is very often no way to determine which method makes more sense for a particular case. To choose the ideal MCDA method, it is necessary to consider input and output information (Ishizaka and Nemery, 2013).

4.3.1.3 MCDA Stages

Traditionally, MCDA application is divided into the eight-step process:

1. Definition of decision context
2. Identification of options to be appraised
3. Identify criteria and objectives
4. Scoring
5. Weighting
6. Scoring & Weighting assessment
7. Results Examination
8. Sensitivity Analysis (Multi-Criteria Analysis, 2009)

4.3.2 Selection of the Optimum DBMS Using MCDA

The MCDA was conducted to choose an ideal DBMS solution for the Georgian Bank. In the case of the selection of the optimum DBMS for the Georgian Bank, the MCDA consisted of:

1. Definition of the Decision Context and Goals
2. Identification of the Options to be Appraised
3. Selection of the MCDA Method
4. Criteria Identification
5. Criteria Scoring
6. Weight Derivation
7. Results Examination

4.3.2.1 Definition of the Decision Context and Goals

The Georgian Bank decided to build a data warehouse solution. To do this, it was needed to pick the best database management software on which the solution would function. As it will be discussed in following sections, there were many DBMS available on the market. Also, there were also many criteria to consider when choosing the optimum DBMS. The goal of this MCDA was to choose the best DBMS for the Georgian Bank and to provide alternative for this best option based on particular criteria. From the perspective of the problem type, this MCDA was supposed to answer questions regarding the choice and ranking problems.

4.3.2.2 Identification of the options to be appraised

There is a number of database management systems on the market offered by a number of vendors. High rated DBMS are those provided by Oracle, Teradata, Cloudera, Microsoft, IBM, SAP, MongoDB, Amazon, Infobright, and many other vendors (Edjali and Beyer, 2016). This wide range of options was reduced to only four options because the other options did not meet the initial requirements such as accessibility of technology, feasibility of solution, and knowledge of the technology from the perspective of the Georgian Bank. The options to be chosen from are briefly introduced below:

Oracle

Oracle is a traditional and well-established provider of DBMS. There is a possibility of complete software and hardware package - Oracle Database 12c, the Oracle Exadata Database Machine, a Cloud solution, and others. Oracle strong points are performance, feature, and stability. There is also an option to enhance the capability of Oracle Database with Hadoop capabilities. As far as the disadvantages are concerned, they are mostly concerned with initial and maintenance costs (Edjali and Beyer, 2016).

Microsoft

Microsoft has Microsft SQL Server 2016 in its DBMS portfolio. Microsoft's main strengths are cloud and self-service capabilities, security, scalability, and familiarity. It is important to mention that the price is lower compared to the Oracle solution. On the other hand, recovery functionality, infrastructure issues, and higher maintenance costs play against this solution.

MongoDB

MongoDB offers and open source DBMS solution whose main strengths are operational data processing, operational analytics, tuning options and friendly user interface. However, this solution has also its disadvantages. MongoDB can be hardly considered as an enterprise analytics platform because it lacks important data warehouse functions including weak procedural extensions. There are also issues related to lock management, authorization, replication and network-attached storage. Besides, developers are not willing to learn this platform due to its immaturity.

Amazon Web Services (AWS)

Amazon Web Service offers Amazon Redshift as its DBMS solution which is a data warehouse service in the cloud. AWS is considered as a leading cloud data warehouse platform-as-a-service provider. It is mainly driven by the acceptance of the cloud, flexibility, and both technical and financial standpoint. As far as the disadvantages are concerned, AWS is a pure cloud vendor. Its solution lacks the support of the hybrid cloud-and-on-premises data warehousing combinations. Also, clients using services of this

vendor reported limitations related to the expectation of the complex DBMS tool (Edjali and Beyer, 2016).

4.3.2.3 Selection of the MCDA Method

In order to achieve the projected goals and to find the best DBMS solution it was needed to pick an appropriate MCDA method. Considering the particular case of the Georgian Bank, the method called PAPRIKA was chosen. It stands for "Potentially All Pairwise RanKings of all possible Alternative". This method is employed to establish criteria weights and to find the optimum DBMS. It is implemented using the MCDA software 1000minds (1000minds). Paprika is method that uses a concept of MCDA or conjoint analysis for establishing decision-makers' preferences through using pairwise rankings of alternatives. PAPRIKA requires a decision-maker to trade-off one characteristics for the other as the figure shows.

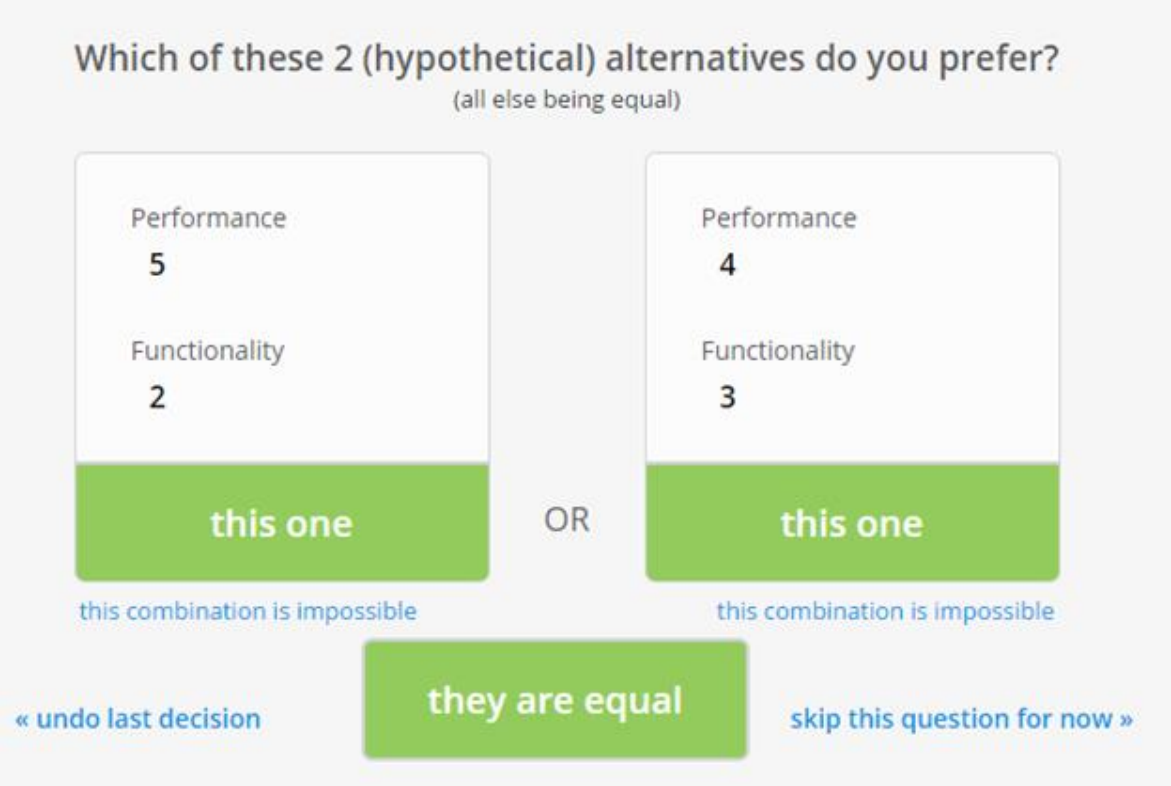


Figure 18 Pairwise Ranking of Alternatives **Source: 1000minds**

Decision-makers express a preference by choosing between two combinations. The software automatically changes the order of the trade-off questions for each survey. This

strategy of swapping the order of questions helps in reducing or eliminating the potential order biases.

On of powerful features of PAPRIKA is it ability in surveying any number of criteria and levels; as these numbers increase, the number of potential alternatives (combinations) increases exponentially. If there are n possible alternative, there are n*(n-1)/2 pairwise rankings. For example, for a value model with five criteria and five categories within each criterion, therefore 55=3125 alternatives, there are 3125*3124/2 = 4,881,250 pairwise rankings. The PAPRIKA method largely reduces the number of selection the decision-maker has to make by reducing dominant pairwise comparisons and use the transitivity feature to implicitly respond to other questions. Domination occurs when a decision is not required for certain alternatives due to the high rate of some alternatives in comparison with others. Then, the "undominated" pairs are to be analyzed by the software. The undominated pair occurs when one alternative has at least one criterion with higher rate and a least one criterion with lower rate in comparison with other alternatives. The software eliminates all the redundant choices when comparing two undominated pairs via transitivity. For example, if choice A is ranked higher than choice B and choice B is higher than choice C, then by transitivity, choice A is ranked higher than choice C. After the two choices, the third choice becomes redundant. Then the software progress in selecting another choice and the process continues until all undominated pairs processed and ranked (Hansen and Ombler, 2008).

4.3.2.4 Criteria Identification

With regard to the Georgian Bank requirements, the project goals, and the objectives of the MCDA, it was needed to set up criteria of DBMS selection. As a result of number of interviews with people in charge of the data warehouse project from the bank side, the following criteria have been set:

Cost

This criterion represents overall DBMS setup and maintenance hardware and software costs, including licensing costs.

Performance

This criterion represents the ability of the database to handle workload which includes ad-hoc queries, analysis, utilities and systems commands. It also represents optimization options.

Functionality

This criterion reflects the functionality of DBMS, such as SQL procedural extension, in-memory support, internal task-scheduling, partitioning, cloud service, advanced backup, big-data storage support and others.

Viability

This criterion represents the Georgian Bank's experience with the DBMS, including the experience with the vendor's products and staff training and ease of hiring.

4.3.2.5 Criteria Scoring

The next step in MCDA was to score the chosen DBMS options in terms of the criteria analyzed in the previous section. This scoring was done considering the features of the DBMS options and with regard to the requirements of the Georgian Bank. The table 1 provides the scoring of the DBMS options.

Vendor	Cost	Performance	Functionality	Viability
Oracle	1	5	5	5
Microsoft	2	4	5	4
AWS	4	3	3	2
MongoDB	5	1	2	1

Table 1: Criteria Scoring

Source: Own Input

To explain the scoring, each vendor could reach the score from one (least favorable) to five (most favorable) in four categories - cost, performance, functionality, and viability. The

vendor reaching the most favorable scoring is regarded as the benchmark for others. This scoring is based on experience with DBMS solution, advice of an experienced expert, and Gartner analysis (Edjali and Beyer, 2016). It does not reflect the overall quality of the products in general, but in this particular cases. The exact explanation of values for each criterion is provided below.

Cost

1= The worst in terms of costs, the DBMS can be regarded as expensive.

5= The best in terms of costs, the DBMS can be regarded as cheap.

Performance

1= The worst in terms of performance, the DBMS can be regarded as a bad performer.

5= The best in terms of performance, the DBMS can be regarded as a good performer.

Functionality

1= The worst in terms of functionality, the DBMS is poor in functions.

5= The best in terms of functionality, the DBMS is rich in functions.

Viability

1= The worst in terms of viability, the Georgian Bank does not have any experience with the vendor's DBMS and its employees have no experience with this DBMS.

5= The best in terms of viability, the Georgian Bank has rich experience with the vendor's DMBS and its customers have rich experience with this DBMS.

4.3.2.6 Weight Derivation

The criteria and their scores were put into the MCDA software (1000minds) to derive weights and the ranking of the analyzed options, providing the most and the least favorable options. These criteria and their scores were evaluated by an individual working at the Georgian Bank who directly participate in the data warehouse implementation. This individual had to express his preference by ranking, prioritizing and choosing between alternatives (figure 18). The list of undominated pairs that needed to be figured out by the MCDA software is attached in the Appendix A.

When the assessment was done, the weights for every criteria score were computed and provided by the MCDA software as the figure 19 depicts:

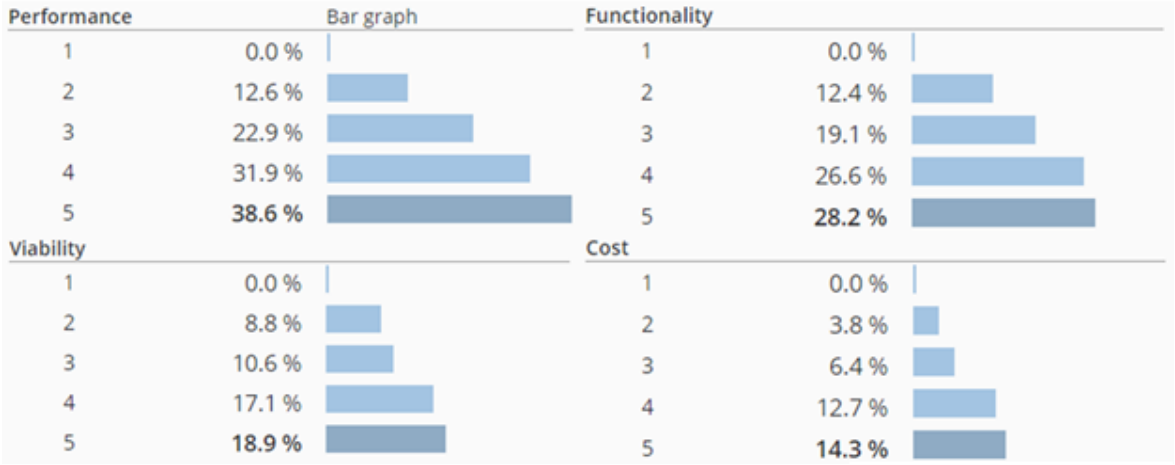


Figure 19: Criteria Weights

Source: 1000minds

After the weights for the scores of particular options were assigned, it was possible to compute total scores for each vendor. It was done by addition of scores with particular weights.

As it can be seen, Oracle scored best with almost 86%, followed by Microsoft with 81.1%, AWS with 63.4%, and MongoDB 26.7%.

Vendor	Cost	Performance	Functionality	Viability	Total Score	Rank
Oracle	1	5	5	5	85.7%	1st
Microsoft	2	4	5	4	81.1%	2nd
AWS	4	3	3	2	63.4%	3rd
MongoDB	5	1	2	1	26.7%	4th

Table 2: Total Score and Ranking

Source: Own Input

4.3.2.7 Results Examination

The MCDA analysis using the PAPRIKA method indicated that Oracle's solution Oracle Database 12c is the most favorable option. This option took advantage of the fact that the Georgian Bank preferred, as the weights indicate, the best performing and most functional option which are the main strengths of the Oracle solution. Microsoft's SQL Server ranked on the second place. It scored better in the cost section which, unfortunately, was not the main preference of the Georgian Bank. AWS's cloud solution seemed as a balanced solution with lower costs compared to the vendors topping the list. However, it suffered from a limited performance and functionality. MongoDB, the only open source solution in this selection, ranked on the last place. Even though it was the best solution in the cost area. Poor performance in all other factors, that turned out to be more important than costs in this case, disqualified this DBMS by a significant margin. The Oracle Database 12c was chosen as the optimum DBMS based on this MCDA.

5 Results and Discussion

The results and discussion section is divided into two parts. The first part provides and discuss results for the theoretical part. The second part focuses on the analytical part.

The theoretical section managed to achieve its main goal which was to introduce and analyze key data warehousing concepts with the special focus on the data warehouse architecture and implementation process. It revealed a number of interesting facts. It was found out that the data warehousing field is not very consistent about the definition of its key terms. There are many terms that have multiple meanings and it is very challenging to define them consistently. A very good example is a definition of the business intelligence. It can be either defined as an umbrella term for all relevant processes and technologies including data integration, master data management, data warehousing, performance management, reporting, analytics and dashboards or it can be also defined as a top layer of a data warehouse. The same applies for the definition of the data warehouse. The data warehouse can be defined as a complex environment or just as a part of the environment (repository). There are also many authors that call the same components of the data warehouse (environment) architecture differently which contributes to further ambiguity. The theoretical part also revealed that there are two major approaches to the data warehouse architecture. The first highlights the importance of the dimensional modelling (Kimball) and the second promotes the third-normal form (Inmon). Every approach suits different purposes and none of them can be used as the ultimate data warehouse architecture. However, when looking for relevant information concerning the data warehouse architecture it very often seemed as a religion war between the supporters believing that their approach is superior to the other one. This resulted in two streams of supporters promoting their own terminology. It was also analyzed that there are two main approaches to the system development life cycle (SDLC), iterative and the waterfall. It was found out that every type of SDLC has its advantages and disadvantages. There is no ideal SDLC because it rather depends on the development circumstances.

As far as the analytical part is concerned, it had three main objectives:

- To introduce the Georgian Bank and its business case
- To analyze the implementation process of the data warehouse

- To select the optimum database management solution for the Georgian Bank.

All these goals were successfully achieved. It was managed to introduce the Georgian Bank from the perspective of its history, strategy, structure, and financial situations. Also the reasons that led the Georgian Bank to build a data warehouse were analyzed. It was found out that the Georgian Bank suffered from inefficient reporting, dependence on certain IT specialist, and poor management practices. These issues prevented the bank from the further growth and reaching its strategic goals. This was also the main reason why the Georgian Bank decided to build the data warehouse that would figure out these issues. It was very interesting to see how minor shortcuts in data management grows in time into a major problem.

The implementation project was conducted using the iterative SDLC because it was needed to minimize the risk of the unknown business environment and changing requirements. The project was scheduled to last for one year and to be delivered in three iterations. The first iteration was supposed to be done in six months, the second in three months, and the third also in three months. Only the first iteration was presented in this thesis as the other iterations are mostly concerned with the enrichment of the first iteration. The iterative SDLC chosen for this project consisted of five steps: analysis & requirements, architecture, design & development, testing, and deployment. During the analysis & requirements gathering step, it was analyzed that the project was feasible so another step of SDLC could begin. The architecture reflected the needs of the Georgian Bank, it consisted of the three main layers – staging area, integrated repository, and data marts. The design itself was a combination of the architectural approaches described in the theoretical part. It was designed to fit the needs of the Georgian Bank. The development step involved the creation of the data warehouse layers. The extract-transfer-load (ETL) process was the most demanding in terms of skills and time since it was very hard to automate. ETL process dealt with movement of data from one layer to another. Also, source system databases were very hard to navigate due to the poor management practices. Testing was a very important part of SDLC as it was necessary to make sure that all the data warehouse components are validated. The most complicated testing part was related to the ETL process again. Once all the components of the data warehouse were validated, the

deployment started. All the components were put into the production to see how they work and how they can be improved further iterations.

The last goal was to choose the optimum database management software for the Georgian Bank. The selection was done using Multi-Criteria Decision Analysis and the method called "Potentially All Pairwise RanKings of all possible Alternative" (PAPRIKA). PAPRIKA is based on assigning weights to options and criteria based on user preferences. This method helped to decide between four DBMS – Microsoft, Oracle, Amazon, and MongoDB. PAPRIKA recommended Oracle Database 12c as the optimum solution for the Georgian Bank despite high costs of this solution. The Oracle solution won due to the Georgian Bank's preferences which were mostly concerned with strong performance and functionality. Surprisingly, the cheapest solution MongoDB ranked on the last place. It is important to emphasize that this evaluation does not reflect the overall quality of DBMS. Weights play the major role in this evaluation.

6 Conclusion

This thesis was divided into two main sections. Each section had its objectives. The theoretical part aimed to investigate the theoretical framework related to data warehousing with emphasis on the data warehouse architecture and the implementation process. The second section, analytical, focused aimed to achieve three objectives. The first goal was to introduce the particular case of the selected company in the banking sector. The second goal was to perform an analysis of the data warehouse design and implementation. Finally, the last goal was to identify the optimum database management software.

Speaking of the objective that was set for the theoretical part, it was managed to investigate and explain the theoretical framework related to data warehousing with emphasis on the architecture and implementation process. To reach this goal, it was necessary to conduct a thorough literature review because there are many different perspectives on the data warehousing issues, particularly on the architecture and the implementation process. The data warehousing is a very broad field and it had to be decided very carefully what to include into the theoretical section. The successful completion of this goal allowed the

author of this thesis to try to achieve the goals of the analytical part because it provided him with the required knowledge.

All the goals of the analytical section were achieved as well. The company as well as the particular business case of the Georgian Bank were introduced and analyzed. It was necessary to analyze many aspects of the Georgian Bank, including the history, strategy, financial position, goals and structure. Besides, the financial situation was analyzed. This part also explained why the Georgian Bank decided to build a data warehouse in the first place. It was very essential to achieve this goal because it created the context for the further analysis. The second goal of the analytical part involved the design and implementation process of the data warehouse for the Georgian Bank. In order to achieve this goal it was needed to take full advantage of the strong theoretical background from the theoretical part. After, the fully functional data warehouse solution was delivered by applying the iterative methodology that was closely described in the theoretical section. However, there is one fact to remind. Even though the project was delivered it got delayed by one month. Finally, the last goal was reached as well. The optimum database management software was identified using the multi-criteria decision analysis (MCDA). The MCDA method called “PAPRIKA” was used to assess weights of criteria. The best solution as well as the overall ranking were provided.

7 References

7.1 Books

ADAMSON, Christopher. *The complete reference star schema*. New York: McGraw-Hill, 2010. ISBN 978-007-1744-331.

ASHDOWN, Lance a Tom KYTE. *Oracle Database Concepts 12c*. 2015. ISBN E41396-13.

DATE, C. J. *An introduction to database systems*. 8th ed. Boston: Pearson/Addison Wesley, 2004. ISBN 03-211-9784-4.

EDJALI, Roxane a Mark A. BEYER. *Magic Quadrant for Data Warehouse and Data Management Solutions for Analytics*. Gartner, 2016. ID:G00275472.

ELEARN. *Making Sense of Data and Information*. United Kingdom: Taylor & Francis, 2013. ISBN 9781136386664.

FRANKS, Patricia C. *Records and Information Management*. Chicago: Neal-Schuman, an imprint of the American Library Association, 2013. ISBN 978-155-5709-105.

HAAN, Lex de., Tim GORMAN, Inger. JORGENSEN a Melanie. CAFFREY. Beginning Oracle SQL: for Oracle database 12c. Third edition. Berkeley, California: Apress, 2014. ISBN 9781430265566.

HANSEN, Paul a Franz OMBLER. A new method for scoring additive multi-attribute value models using pairwise rankings of alternatives. *Journal of Multi-Criteria Decision Analysis*. 2008, (15), 87-107. DOI: 10.1002/mcda.428.

HERNANDEZ, Michael J. *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. Second edition. Massachusetts: Addison-Wesley, 2003. ISBN 0-201-75284-0.

INMON, William H. *Building the data warehouse*. 4th ed. Indianapolis, Ind.: Wiley, c2005. ISBN 07-645-9944-5.

ISHIZAKA, Alessio a Philippe NEMERY. *Multi-Criteria Decision Analysis: Methods and Software*. Wiley, 2013. ISBN 978-1-119-97407-9.

KIMBALL, Ralph a Margy ROSS. *The data warehouse toolkit: the definitive guide to dimensional modeling*. 3rd ed. Indianapolis, Ind.: Wiley, c2013. ISBN 978-1-118-73219-9.

KELLEY, Jüris. *Knowledge nirvana: achieving the competitive advantage through enterprise content management and optimizing team collaboration*. Fairfax, VA: Xulon Press, 2002. ISBN 9781591601081.

LANNING, Scott. *Reference and instructional services for information literacy skills in school libraries*. Third edition. Santa Barbara, CA: Libraries Unlimited, an imprint of ABC-CLIO, LLC, 2014. ISBN 9781610696715.

LIEBOWITZ, Jay. *Strategic intelligence: business intelligence, competitive intelligence, and knowledge management*. Boca Raton, FL: Auerbach Publications, 2006. ISBN 978-084-9398-681.

LIGHTSTONE, Sam, Toby J. TEOREY a Tom NADEAU. *Physical database design: the database professional's guide to exploiting indexes, views, storage, and more*. Boston: Morgan Kaufmann/Elsevier, c2007. Morgan Kaufmann series in data management systems. ISBN 978-0-12-369389-1.

LINSTEDT, Daniel & Michael OLSCHIMKE. *Building a Scalable Data Warehouse with Data Vault 2.0*. 225 Wyman Street, Waltham, MA 02451, USA: Elsevier, 2016. ISBN 978-0-12-802510-9.

LOTZ, Mary. *Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?* [online]. 2013 [cit. 2016-11-06]. Available from: http://www.seguetech.com/waterfall-vs-agile-methodology/

Multi-criteria analysis: a manual. London: Communities and Local Government Publications, 2009. ISBN 978-1-4098-1023-0.

PONNIAH, Paulraj. *Data Warehousing Fundamentals a Comprehensive Guide for IT Professionals*. Hoboken, NJ: John Wiley, 2001. ISBN 978-047-1463-894.

RAINARDI, Vincent. *Building a data warehouse with examples in SQL Server*. Berkeley, CA: Apress; Distributed to the book trade worldwide by Springer-Verlag New York, c2008. ISBN 1590599314.

RAINE, L., & WELLMAN, B. *Networked. The new social operating system*. Cambridge: MIT Press, 2012. ISBN: 9780262017190.

RAMKLASS, Roopesh. *OCA Oracle Database 12c: SQL fundamentals I exam guide (Exam 1Z0-061)*. Second edition. New York: McGraw-Hill Education, 2014. ISBN 9780071820288.

ROWLEY, Jennifer a Richard HARTLEY. *Organizing Knowledge: An Introduction to Managing Access to Information*. 4th edition. Routledge, 2008. ISBN 978-0754644316.

RUMBAUGH, James, Ivar JACOBSON a Grady BOOCH. *The unified modeling language reference manual*. 2nd ed. Boston: Addison-Wesley, c2005. Addison-Wesley object technology series. ISBN 03-212-4562-8.

TEOREY, Toby J. *Database modeling and design: logical design*. 5th ed. Boston: Morgan Kaufmann Publishers, c2011. Morgan Kaufmann series in data management systems. ISBN 9780123820204.

WALLACE, Danny P. *Knowledge Management: Historical and Cross-Disciplinary Themes* (Libraries Unlimited Knowledge Management Series). 2nd edition. Libraries Unlimited, 2007. ISBN 978-1591585022.

7.2 Online Sources

1000minds [online]. 2016 [cit. 2016-11-30]. Available From: https://www.1000minds.com/

90% OF WORLD'S DATA CREATED IN LAST TWO YEARS. *ITEUROPA* [online]. 2015 [cit. 2016-11-30]. Available From: http://www.iteuropa.com/?q=90-worlds-data-created-last-two-years

An Enterprise Architect's Guide to Big Data: Reference Architecture Overview. Oracle [online]. 2016 [cit. 2016-07-11]. Available on: http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf

Business Intelligence: IT Glossary. Gartner [online]. [cit. 2016-06-29]. Available on: http://www.gartner.com/it-glossary/business-intelligence-bi/

CEBOTAREAN, Elena. *Business Intelligence. Scientific Papers: Journal of Knowledge Management, Economics, and information Technology* [online]. 2011, , 12 [cit. 2016-06-29]. ISSN 2069-5934. Available on: http://www.scientificpapers.org/economics/business-intelligence/

Cloud computing. *Webopedia* [online]. 2016 [cit. 2016-11-05]. Available from: http://www.webopedia.com/TERM/C/cloud_computing.html

Data Vault Basics. DanLinstedt [online]. [cit. 2016-11-05]. Available from: http://danlinstedt.com/solutions-2/data-vault-basics/

Data Warehouse: *The Choice of Inmon versus Kimball*. *Scribd* [online]. IAS Inc. [cit. 2016-10-30]. Available from: https://www.scribd.com/doc/253618546/080827Abramson-Inmon-vs-Kimball

Data Warehousing Concepts. *1Keydata* [online]. [cit. 2016-11-12]. Available From: http://www.1keydata.com/datawarehousing/concepts.html

Database management system (DBMS). *Searchsqlserver: Techtarget* [online]. [cit. 2016-11-13]. Available From: http://searchsqlserver.techtarget.com/definition/database-management-system

HENSCHEN, Doug. 5 Analytics, BI, Data Management Trends For 2015. In: *Information Week* [online]. 2015 [cit. 2016-11-05]. Available from: http://www.informationweek.com/big-data/big-data-analytics/5-analytics-bi-data-management-trends-for-2015/a/d-id/1318551

Index (IDX). *Techopedia* [online]. [cit. 2016-11-13]. Available From: https://www.techopedia.com/definition/1210/index-idx-database-systems

Inmon vs. Kimball: Which approach is suitable for your data warehouse? ComputerWeekly [online]. 2012 [cit. 2016-11-17]. Available From: http://www.computerweekly.com/tip/Inmon-vs-Kimball-Which-approach-is-suitable-for-your-data-warehouse

LANE, Paul a Padjama POTINENI. *Oracle Database Data Warehousing Guide: 12c Release 1 (12.1)* [online]. 2014 [cit. 2016-11-05]. ISBN E41670-08. Available from: https://docs.oracle.com/database/121/DWHSG/E41670-08.pdf

MARVIN, Rob. 10 Business Intelligence Trends for 2016. In: *PCMAG* [online]. 2015 [cit. 2016-11-05]. Available from: http://www.pcmag.com/article2/0,2817,2496370,00.asp

SELECTING A DEVELOPMENT APPROACH. In: *CMS: Office of Information Services* [online]. 2005 [cit. 2016-11-06]. Available from: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

Sequence Numbers. Microsoft [online]. 2016 [cit. 2016-11-05]. Available from: https://msdn.microsoft.com/en-us/library/ff878058.aspx

Timestamp. *Microsoft* [online]. 2016 [cit. 2016-11-05]. Available from: https://msdn.microsoft.com/en-us/library/ms182776(v=SQL.90).aspx

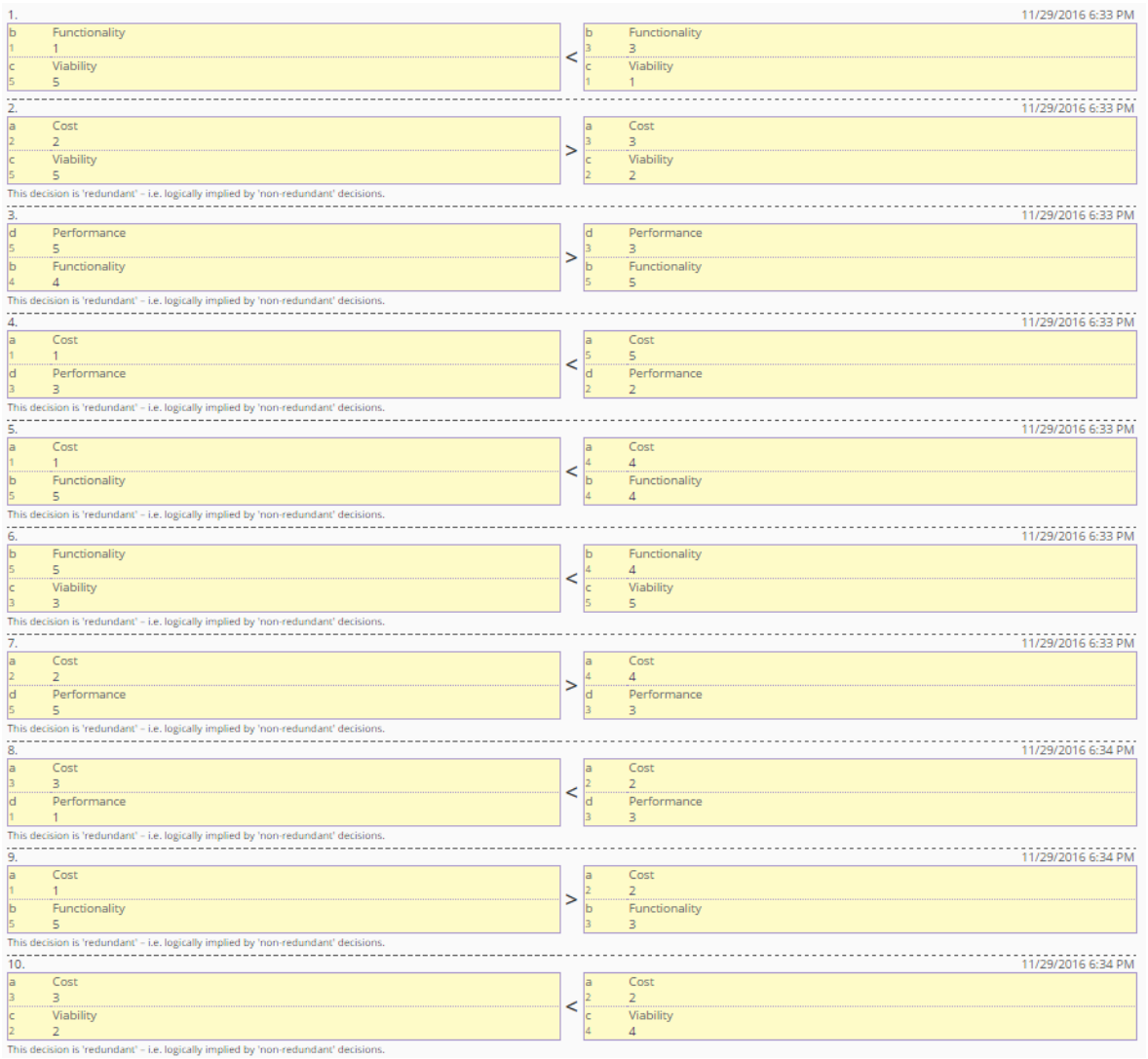
VAN LOON, Ronald. What Is the Future of Data Warehousing? In: *Data Science Central* [online]. 2016 [cit. 2016-11-05]. Available from: http://www.datasciencecentral.com/profiles/blogs/what-is-the-future-of-data-warehousing

View. *Techopedia* [online]. [cit. 2016-11-13]. Available From: https://www.techopedia.com/definition/25126/view-databases

What is Project Management? *PMI: Project Management Institute* [online]. 2016 [cit. 2016-11-06]. Available from: https://www.pmi.org/about/learn-about-pmi/what-is-project-management

8 Appendix

8.1 Appendix A: Paprika Audit Report



1. 11/29/2016 6:33 PM

b	Functionality			b	Functionality
1	1			3	3
c	Viability	<		c	Viability
5	5			1	1

2. 11/29/2016 6:33 PM

a	Cost			a	Cost
2	2			3	3
c	Viability	>		c	Viability
5	5			2	2

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

3. 11/29/2016 6:33 PM

d	Performance			d	Performance
5	5			3	3
b	Functionality	>		b	Functionality
4	4			5	5

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

4. 11/29/2016 6:33 PM

a	Cost			a	Cost
1	1			5	5
d	Performance	<		d	Performance
3	3			2	2

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

5. 11/29/2016 6:33 PM

a	Cost			a	Cost
1	1			4	4
b	Functionality	<		b	Functionality
5	5			4	4

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

6. 11/29/2016 6:33 PM

b	Functionality			b	Functionality
5	5			4	4
c	Viability	<		c	Viability
3	3			5	5

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

7. 11/29/2016 6:33 PM

a	Cost			a	Cost
2	2			4	4
d	Performance	>		d	Performance
5	5			3	3

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

8. 11/29/2016 6:34 PM

a	Cost			a	Cost
3	3			2	2
d	Performance	<		d	Performance
1	1			3	3

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

9. 11/29/2016 6:34 PM

a	Cost			a	Cost
1	1			2	2
b	Functionality	>		b	Functionality
5	5			3	3

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

10. 11/29/2016 6:34 PM

a	Cost			a	Cost
3	3			2	2
c	Viability	<		c	Viability
2	2			4	4

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

11. 11/29/2016 6:34 PM

a	Cost		a	Cost	
5	5	<	4	4	
c	Viability		c	Viability	
2	2		3	3	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

12. 11/29/2016 6:34 PM

a	Cost		a	Cost	
4	4	>	5	5	
d	Performance		d	Performance	
4	4		3	3	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

13. 11/29/2016 6:34 PM

a	Cost		a	Cost	
4	4	<	1	1	
d	Performance		d	Performance	
1	1		3	3	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

14. 11/29/2016 6:34 PM

d	Performance		d	Performance	
4	4	>	2	2	
c	Viability		c	Viability	
1	1		3	3	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

15. 11/29/2016 6:34 PM

a	Cost		a	Cost	
3	3	>	5	5	
b	Functionality		b	Functionality	
5	5		3	3	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

16. 11/29/2016 6:35 PM

d	Performance		d	Performance	
3	3	<	5	5	
c	Viability		c	Viability	
3	3		2	2	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

17. 11/29/2016 6:35 PM

d	Performance		d	Performance	
3	3	>	2	2	
c	Viability		c	Viability	
3	3		4	4	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

18. 11/29/2016 6:35 PM

a	Cost		a	Cost	
3	3	>	2	2	
c	Viability		c	Viability	
2	2		3	3	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

19. 11/29/2016 6:35 PM

d	Performance		d	Performance	
1	1	<	5	5	
c	Viability		c	Viability	
5	5		2	2	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

20. 11/29/2016 6:35 PM

d	Performance		d	Performance	
2	2	>	3	3	
c	Viability		c	Viability	
3	3		1	1	

This decision is 'redundant' – i.e. logically implied by 'non-redundant' decisions.

21. 11/29/2016 6:35 PM

a	Cost		a	Cost	
1	1	>	5	5	
d	Performance		d	Performance	
3	3		1	1	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

22. 11/29/2016 6:35 PM

d	Performance		d	Performance	
5	5	<	2	2	
b	Functionality		b	Functionality	
1	1		4	4	

23. 11/29/2016 6:36 PM

d	Performance		d	Performance	
2	2	<	3	3	
c	Viability		c	Viability	
5	5		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

24. 11/29/2016 6:37 PM

d	Performance		d	Performance	
5	5	<	4	4	
b	Functionality		b	Functionality	
1	1		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

25. 11/29/2016 6:37 PM

a	Cost		a	Cost	
3	3	>	4	4	
c	Viability		c	Viability	
4	4		1	1	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

26. 11/29/2016 6:37 PM

a	Cost		a	Cost	
4	4	>	1	1	
c	Viability		c	Viability	
2	2		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

27. 11/29/2016 6:37 PM

a	Cost		a	Cost	
1	1	<	4	4	
b	Functionality		b	Functionality	
5	5		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

28. 11/29/2016 6:37 PM

a	Cost		a	Cost	
2	2	>	3	3	
c	Viability		c	Viability	
3	3		1	1	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

29. 11/29/2016 6:38 PM

b	Functionality		b	Functionality	
5	5	>	3	3	
c	Viability		c	Viability	
2	2		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

30. 11/29/2016 6:38 PM

d	Performance		d	Performance	
2	2	>	1	1	
b	Functionality		b	Functionality	
1	1		2	2	

31. 11/29/2016 6:38 PM

a	Cost		a	Cost	
5	5	>	1	1	
b	Functionality		b	Functionality	
2	2		4	4	

32. 11/29/2016 6:39 PM

d	Performance		d	Performance	
1	1	>	2	2	
b	Functionality		b	Functionality	
4	4		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

33. 11/29/2016 6:39 PM

d	Performance		d	Performance	
3	3	<	5	5	
b	Functionality		b	Functionality	
3	3		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

34. 11/29/2016 6:39 PM

b	Functionality		b	Functionality	
5	5	<	2	2	
c	Viability		c	Viability	
1	1		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

35. 11/29/2016 6:39 PM

a	Cost		a	Cost	
1	1	>	3	3	
c	Viability		c	Viability	
4	4		3	3	

36. 11/29/2016 6:40 PM

b	Functionality		b	Functionality	
2	2	<	3	3	
c	Viability		c	Viability	
3	3		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

37. 11/29/2016 6:40 PM

a	Cost		a	Cost	
4	4	<	3	3	
c	Viability		c	Viability	
3	3		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

38. 11/29/2016 6:40 PM

d	Performance		d	Performance	
4	4	>	2	2	
b	Functionality		b	Functionality	
3	3		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

39. 11/29/2016 6:42 PM

d	Performance		d	Performance	
1	1	<	3	3	
b	Functionality		b	Functionality	
5	5		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

40. 11/29/2016 6:42 PM

d	Performance		d	Performance	
5	5	>	3	3	
c	Viability		c	Viability	
2	2		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

41. 11/29/2016 6:42 PM

d	Performance		d	Performance	
3	3	<	4	4	
b	Functionality		b	Functionality	
3	3		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

42. 11/29/2016 6:42 PM

a	Cost		a	Cost	
5	5	>	2	2	
c	Viability		c	Viability	
3	3		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

43. 11/29/2016 6:44 PM

a	Cost		a	Cost	
1	1	<	5	5	
d	Performance		d	Performance	
4	4		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

44. 11/29/2016 6:44 PM

a	Cost		a	Cost	
4	4	>	2	2	
c	Viability		c	Viability	
3	3		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

45. 11/29/2016 6:45 PM

a	Cost		a	Cost	
5	5	<	3	3	
c	Viability		c	Viability	
3	3		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

46. 11/29/2016 6:45 PM

a	Cost		a	Cost	
5	5	<	3	3	
d	Performance		d	Performance	
3	3		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

47. 11/29/2016 6:45 PM

a	Cost		a	Cost	
4	4	<	2	2	
c	Viability		c	Viability	
1	1		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

48. 11/29/2016 6:45 PM

b	Functionality		b	Functionality	
5	5	>	2	2	
c	Viability		c	Viability	
2	2		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

49. 11/29/2016 6:45 PM

d	Performance		d	Performance	
5	5	>	4	4	
c	Viability		c	Viability	
3	3		4	4	

50. 11/29/2016 6:46 PM

d	Performance		d	Performance	
2	2	<	4	4	
b	Functionality		b	Functionality	
3	3		1	1	

50. 11/29/2016 6:46 PM

d	Performance		d	Performance	
2	2	<	4	4	
b	Functionality		b	Functionality	
3	3		1	1	

51. 11/29/2016 6:46 PM

d	Performance		d	Performance	
3	3	>	2	2	
b	Functionality		b	Functionality	
2	2		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

52. 11/29/2016 6:46 PM

a	Cost		a	Cost	
2	2	<	3	3	
c	Viability		c	Viability	
5	5		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

53. 11/29/2016 6:46 PM

d	Performance		d	Performance	
3	3	>	4	4	
b	Functionality		b	Functionality	
5	5		3	3	

54. 11/29/2016 6:46 PM

d	Performance		d	Performance	
1	1	<	2	2	
c	Viability		c	Viability	
4	4		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

55. 11/29/2016 6:46 PM

a	Cost		a	Cost	
2	2	>	4	4	
d	Performance		d	Performance	
4	4		3	3	

56. 11/29/2016 6:47 PM

d	Performance		d	Performance	
3	3	>	2	2	
c	Viability		c	Viability	
1	1		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

57. 11/29/2016 6:48 PM

d	Performance		d	Performance	
2	2	<	3	3	
c	Viability		c	Viability	
5	5		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

58. 11/29/2016 6:48 PM

a	Cost		a	Cost	
3	3	>	2	2	
b	Functionality		b	Functionality	
4	4		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

59. 11/29/2016 6:48 PM

a	Cost		a	Cost	
5	5	>	2	2	
d	Performance		d	Performance	
4	4		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

60. 11/29/2016 6:48 PM

a	Cost		a	Cost	
2	2	>	3	3	
b	Functionality		b	Functionality	
3	3		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

60. 11/29/2016 6:48 PM

a	Cost		a	Cost	
2	2	>	3	3	
b	Functionality		b	Functionality	
3	3		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

61. 11/29/2016 6:48 PM

a	Cost		a	Cost	
3	3	<	1	1	
b	Functionality		b	Functionality	
2	2		3	3	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

62. 11/29/2016 6:48 PM

a	Cost		a	Cost	
4	4	>	1	1	
d	Performance		d	Performance	
1	1		2	2	

63. 11/29/2016 6:49 PM

a	Cost		a	Cost	
5	5	<	3	3	
c	Viability		c	Viability	
2	2		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

64. 11/29/2016 6:51 PM

d	Performance		d	Performance	
3	3	>	2	2	
b	Functionality		b	Functionality	
3	3		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

65. 11/29/2016 6:51 PM

b	Functionality		b	Functionality	
4	4	>	5	5	
c	Viability		c	Viability	
3	3		2	2	

66. 11/29/2016 6:51 PM

a	Cost		a	Cost	
5	5	<	4	4	
c	Viability		c	Viability	
4	4		5	5	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

67. 11/29/2016 6:52 PM

b	Functionality		b	Functionality	
2	2	>	3	3	
c	Viability		c	Viability	
2	2		1	1	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

68. 11/29/2016 6:52 PM

a	Cost		a	Cost	
2	2	<	4	4	
c	Viability		c	Viability	
2	2		1	1	

69. 11/29/2016 6:52 PM

b	Functionality		b	Functionality	
4	4	>	5	5	
c	Viability		c	Viability	
5	5		4	4	

70. 11/29/2016 6:52 PM

b	Functionality		b	Functionality	
5	5	>	3	3	
c	Viability		c	Viability	
2	2		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

71. 11/29/2016 6:52 PM

d	Performance		d	Performance	
5	5	<	3	3	
c	Viability		c	Viability	
1	1		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

72. 11/29/2016 6:52 PM

d	Performance		d	Performance	
3	3	>	2	2	
c	Viability		c	Viability	
2	2		5	5	

73. 11/29/2016 6:52 PM

a	Cost		a	Cost	
5	5	<	4	4	
b	Functionality		b	Functionality	
4	4		5	5	

74. 11/29/2016 6:52 PM

b	Functionality		b	Functionality	
3	3	>	2	2	
c	Viability		c	Viability	
3	3		4	4	

75. 11/29/2016 6:52 PM

d	Performance		d	Performance	
4	4	>	3	3	
c	Viability		c	Viability	
2	2		4	4	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

76. 11/29/2016 6:52 PM

a	Cost		a	Cost	
3	3	>	4	4	
b	Functionality		b	Functionality	
3	3		2	2	

This decision is 'redundant' - i.e. logically implied by 'non-redundant' decisions.

77. 11/29/2016 6:52 PM

a	Cost		a	Cost	
5	5	>	2	2	
d	Performance		d	Performance	
2	2		3	3	

78. 11/29/2016 6:53 PM

a	Cost		a	Cost	
5	5	<	2	2	
c	Viability		c	Viability	
1	1		3	3	

79. 11/29/2016 6:53 PM

d	Performance		d	Performance	
5	5	<	3	3	
b	Functionality		b	Functionality	
2	2		5	5	

80. 11/29/2016 6:53 PM

a	Cost		a	Cost	
3	3	>	4	4	
c	Viability		c	Viability	
4	4		3	3	