

**CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE**

Faculty of Economics and Management

Informatics

*Department of Information Engineering*



**Diploma Thesis**

**Data Warehouses: Comparison between Apache Hadoop and  
Microsoft implementations on a real life example**

**Author:** Hussein Mohamed Negm  
**Supervisor:** doc. Ing. Vojtěch Merunka, Ph.D.

© 2016 CULS Prague

# DIPLOMA THESIS ASSIGNMENT

Hussein Mohamed Abdelhaq Ragab Negm

Informatics

Thesis title

**Datawarehouse**

---

## **Objectives of thesis**

Design and implement a datawarehouse of a concrete multinational company that operates call centers supporting services for other subjects in area of telecommunications.

## **Methodology**

1. Prefer open-source technology and perform a comparison between open-source and Microsoft platform.
2. Follow approach of CRM (Customer-Relationship-Management)

**The proposed extent of the thesis**

60-80 pages

**Keywords**

MySQL, MS SQL, ETL Processes, CRM, BI

---

**Recommended information sources**

Barry Devlin, Data Warehouse: From Architecture to Implementation, 1996

Krish Krishnan, Data Warehousing in the Age of Big Data, 2013

Lakshman Bulusu, Open Source Data Warehousing and Business Intelligence, 2012

Ralph Kimball, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 2013

---

**Expected date of thesis defence**

2015/16 SS – FEM

**The Diploma Thesis Supervisor**

doc. Ing. Vojtěch Merunka, Ph.D.

**Supervising department**

Department of Information Engineering

Electronic approval: 20. 2. 2016

**Ing. Martin Pelikán, Ph.D.**

Head of department

Electronic approval: 20. 2. 2016

**Ing. Martin Pelikán, Ph.D.**

Dean

Prague on 16. 03. 2016

---

## **Declaration**

I declare that I have worked on my diploma thesis titled "Data Warehouses: Comparison between Apache Hadoop and Microsoft implementations on a real life example" 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

---

Hussein Negm

### **Acknowledgement**

I would like to thank my supervisor doc. Ing. Vojtěch Merunka for his help and guidance during this project. I would also like to thank all my teachers and professors at every educational level for knowledge is accumilative. Special thanks for my family and my wife Esraa for all the support during the time I spent in Czech Republic. Finally, I would like to thank the Czech ministry of foreign affairs for making it all possible through the scholarship

## **Shrnutí**

Firmy produkují stále více dat a hledají nové způsoby, jak je obchodně zhodnotit. S rostoucími objemy dat přichází požadavek na lepší a levnější uchovávání, které by zároveň umožňovalo tato data využívat. Pro tento účel se jako nejhodnější nástroj osvědčily datové sklady. Datové sklady ovšem představují značné personální a finanční náklady. Nabídka technologií pro implementování datových skladů je všestranná. Cílem tohoto projektu je komparativní implementace s použitím dvou technologií, konkrétně Microsoft SQL Server a Apache Hadoop. Tento projekt se věnuje různým fázím vytváření datového skladu; fáze určování požadavků; fáze návrhu a kompaktní porovnání entitně-vztahového modelování s dimenzionálním modelováním a proces budování dimenzionálního modelu na základě prostředků aplikačních dat; fáze extrahování-převádění-zavádění. Poté jsou obě technologie porovnány z hlediska kapacity dat, načítání dat, připojení a dotazování dat. Ze závěrů tohoto projektu vyplývá, že při výběru mezi systémy Microsoft SQL Server a Apache Hadoop je třeba vycházet z potřeb, prostředků a stávajícího ekosystému. Systém Hadoop je vhodný pro větší objemy dat, pro nestrukturované nebo nepravidelné formáty dat a pro případ, že rozpočet nepočítá s licenčními poplatky. Na druhou stranu systém Microsoft SQL Server je vhodnou volbou pro strukturovaná data, když jsou známy očekávané objemy dat a když je zbývající část ekosystému založena na produktech Microsoft. Další fáze tohoto projektu by měly být věnovány novým způsobům efektivnějšího využití systému Hadoop pro menší objemy dat, jako jsou Impala a Spark..

### **Klíčová slova**

Data Warehouse, Microsoft SQL Server, Apache Hadoop, Hive, Dimensional Modeling, HDFS

# **Abstract**

Data is being produced by the firms in ever increasing rates and firms are finding new ways to make use of data to create business value. The generated volumes of data create the need for better and cheaper storage options that allows utilizing the data as well. Data warehouses have emerged as the most appropriate tool for this task. However, data warehouses come with significant costs both human and financial. The pool of technologies for implementing data warehouses is versatile. This project aims to provide a comparative implementation using two of the technologies, namely, Microsoft SQL Server and Apache Hadoop. The project covers the different phases of building a data warehouse; the requirements specification phase; the design phase and a compact comparison between the entity-relation and dimensional modeling design techniques and the process of building a dimensional model based on based on the application data sources; the extract-transform-load phase. The comparison is then made between the two technologies for data capacity, data loading, connectivity and querying data. The project concludes that the decision to choose between Microsoft SQL Server and Apache Hadoop isn't a recommendation for one over the other but should be based on the needs, resources and the existing ecosystem. Hadoop would be the choice for bigger amounts of data, unstructured or irregular data formats, and when the licensing fees are an unaffordable cost. On the other hand, Microsoft SQL Server would make a better choice when the data is structured, the anticipated data volumes are suitable and when the rest of ecosystem is Microsoft based. Future development for this project should cover new ways to make Hadoop more efficient with smaller data volumes like Impala and Spark.

## **Key words**

Data Warehouse, Microsoft SQL Server, Apache Hadoop, Hive, Dimensional Modeling, HDFS

## Table of Contents

1. Introduction.....	1
1.1 Storing Data .....	1
1.2 Data warehouses .....	1
1.3 The difference between a data warehouse and a database .....	3
1.3.1 The purpose.....	4
1.3.2 The scale of operation .....	4
1.3.3 The design fundamentals.....	4
1.4 Introducing the business.....	5
2. Thesis Objective and methodology .....	6
2.1 Objective .....	6
2.2 Methodology .....	7
3. Literature Review.....	8
3.1 Data Warehousing.....	8
3.1.1 Entity-Relationship Modeling.....	10
3.1.2 Dimensional Modeling.....	12
3.2 Apache Hadoop.....	17
3.2.1 YARN .....	19
3.2.2 Hadoop Distributed File System .....	21
3.2.3 Hive.....	23
3.2.4 Why Hadoop?.....	24
4. Project Implementation .....	27
4.1 Business Requirements .....	27
4.1.1 Information Requirements .....	27
4.1.2 Analytical Requirements .....	28
4.1.3 Technical Requirements.....	29
4.2 Analyzing Data Sources.....	30
4.2.1 TD Apro .....	30
4.2.2 TranslatorAdvisor .....	32



4.3 Dimensional Modeling.....	33
4.3.1 Naming Terminology .....	33
4.3.2 Dimensions.....	33
4.3.3 Facts .....	40
4.4 ETL (Extract, Transform, Load) .....	43
4.4.1 Etraction .....	43
4.4.2 Transformation.....	47
4.4.3 Loading .....	49
4.5 Implementation Remarks .....	58
5. Conclusion .....	60
5.1 Apache Hadoop.....	60
5.2 Microsoft SQL Server .....	61
References.....	62

## List of Figures

FIGURE 1: DATA WAREHOUSE DESIGN CYCLE [SOURCE: OWN]	3
FIGURE 2 TYPE 1- TRANSFORMATION [SOURCE: OWN]	11
FIGURE 3 DIMENSION MODEL EXAMPLE [SOURCE: OWN]	12
FIGURE 4 HADOOP UNDERLYING SYSTEM ARCHITECTURE [SOURCE: (APACHE SOFTWARE FOUNDATION, 2016)]	18
FIGURE 5: SIMPLIFIED TDAPRO APPLICATION DATABASE [SOURCE: OWN]	31
FIGURE 6: GEOGRAPHY DIMENSION [SOURCE: OWN]	34
FIGURE 7: CUSTOMER DIMENSION [SOURCE: OWN]	35
FIGURE 8: DATE DIMENSION [SOURCE: OWN]	37
FIGURE 9 LANGUAGE DIMENSION [SOURCE: OWN]	38
FIGURE 10 TRANSLATOR DIMENSION [SOURCE: OWN]	39
FIGURE 11 SALES FACT [SOURCE: OWN]	41
FIGURE 12 DIMENSION MODEL [SOURCE: OWN]	42
FIGURE 13: SSMS MENU [SOURCE: OWN]	44
FIGURE 14: COPY DATABASE FIRST STEP [SOURCE: OWN]	45
FIGURE 15 COPY DESTINATION [SOURCE: OWN]	46
FIGURE 16 DATABASE ONLINE DURING COPY [SOURCE: OWN]	47
FIGURE 17 TRANSFORMATION USING SSIS [SOURCE: OWN]	48
FIGURE 18 IMPORT DATA [SOURCE: OWN]	50
FIGURE 19 LOADING DATA INTO SQL SERVER [SOURCE: OWN]	51
FIGURE 20 INPUT FILE AND COLUMN MAPPINGS [SOURCE: OWN]	52
FIGURE 21 HUE LOGIN SCREEN [SOURCE: OWN]	53
FIGURE 22 HUE FILE BROWSER [SOURCE: OWN]	54
FIGURE 23 HUE NEW DIRECTORY [SOURCE: OWN]	54
FIGURE 24 HUE UPLOAD FILES [SOURCE: OWN]	54
FIGURE 25 HUE: FILES ADDED INTO HDFS [SOURCE: OWN]	55
FIGURE 26 HUE: CREATING A TABLE [SOURCE: OWN]	55
FIGURE 27 HUE: SAMPLE CUSTOMER DATA [SOURCE: OWN]	56

## List of tables

TABLE 1 TYPE-TWO DIMENSION CHANGE [SOURCE: OWN] .....	14
TABLE 2 TYPE-THREE DIMENSION CHANGE [SOURCE: OWN].....	14
TABLE 3 COSTS OF ORACLE DATA WAREHOUSE [SOURCE: OWN] .....	25
TABLE 4 COSTS OF HADOOP DATA WAREHOUSE [SOURCE: OWN].....	25
TABLE 5 FINAL ASSESSMENT [SOURCE: OWN] .....	58

## **1. Introduction**

### **1.1 Storing Data**

Humans have been collecting data in various forms since the dawn of history even before constituting formal writing rules. Samaritans and Ancient Egyptians collected data about astronomy, agriculture and their life events in the form of drawings on the walls of their temples, tombs and temples and later papyrus. Then we moved to hand writing and then printing all the way to the computer age. The amount of data produced has naturally been positively correlated with the ease and speed of the data creation process. Writing a book used to be a several years task and later making copies of this book wasn't a trivial task as well but as we move to a more digital world we generate data at unprecedented rates and in a very diverse variety of forms.

For example, the amount of data on Youtube is estimated to hundreds of millions of hours (Youtube, 2015), Instagram contains billions of images; Google search engine holds records for an ever-growing number of web pages and their content. Even on an individual scale, we now tend to take more pictures with our phones and digital cameras, write more in forms of emails, blog posts or personal websites and store other sorts of professional manuscripts or architectural designs in digital formats. Another striking example is medical research where acquiring data is an expensive process and data comes in huge chunks so accurately storing data is at the core of the research endeavor. For example, data from a single genome sequence is about 200GB (Genomics England, 2014). Data has evolved as one of the most important assets for many businesses, research centers, governments and universities. Data is used to build models to evaluate performance, make plans for the future, develop new products, and decide where to build stores along with many other uses.

### **1.2 Data warehouses**

Storing digital data has passed through various stages from mercury delayed storage moving to different kind of magnetic techniques. The noticeable development is in the capacity and

decrease in the cost of storage. Just 50 years ago IBM rented a computer that can store 2 million digits for 850\$/month (Spicer, et al., 1998). Now, for the same amount of money you can buy a computer that is millions of times faster and bigger in storage capacity while being smaller in size and more power efficient. Data warehouses are the de-facto technology for storing large magnitudes of data for enterprises or for research facilities.

A data warehouse is in essence a relational database designed for more efficient querying and analysis of data rather than transactional integrity. As the name suggests a data warehouse will aggregate all the data from different sources. This aggregation process would be referred to as ETL or Extract-Transfer-Load process; Extracting the data from the sources; Transforming the data to match the data warehouse design; Loading the data into the warehouse to be ready for further use.

The process of designing and implementing a data warehouse is highly structured and of great importance. This is mainly due to the fact that data warehouses are naturally developed to allow for better decision making in the organization and decision are a usually taken at the highest level of the organization and in many cases would reflect considerable results, gains or losses. The interpretation of the term Data Warehouse itself varies widely across the literature. However, in this paper, we will follow the Ralph Kimball understanding of data warehouses, dimensional design and end-to-end role in business intelligence.

In the literature review section, this understanding will be covered in enough depth and contrasted to the other philosophy from Bill Inmon (Inmon, 1992). The process of implementing a data warehouse goes through many phases to ensure correct and efficient implementation. The process starts by collecting the requirements or objectives from the data warehouse as a business intelligence facilitation tool then moves to dimensional modeling then physical design then the design of ETL processes and finally actual implementation on the selected infrastructure. The following diagram gives a better visualization of the process that will be implemented for this project

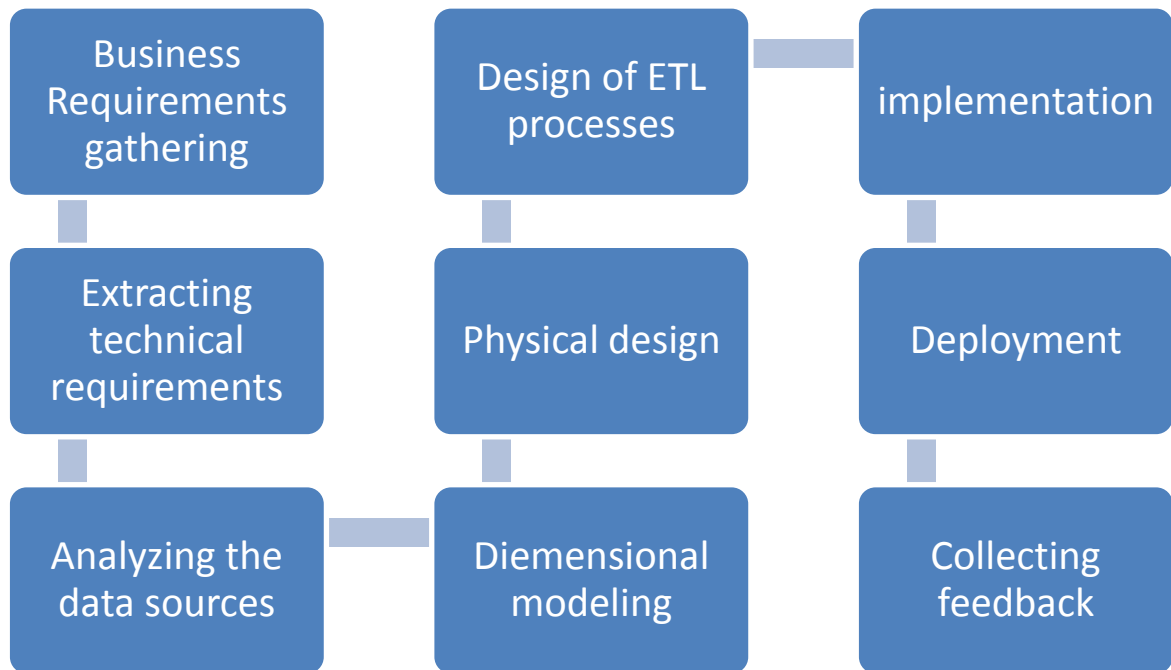


Figure 1: Data Warehouse Design Cycle [source: own]

### 1.3 The difference between a data warehouse and a database

It is a common mistake to confuse databases with data warehouses since they are both used to store data and since many don't get the chance to deal with data warehouses in the course of their digital experience and others consider a data warehouse just a big database or consolidation of multiple databases or data sources. There many noticeable differences that relates to:

- The different purposes of databases and data warehouses.
- The scale of operation.
- The design fundamentals.

### **1.3.1 The purpose**

Databases are the functional unit of almost all the modern digital products it is used to store transactional information regarding the use of the application, the content of the application or the business rules that applies on different cases. In contrary, a data warehouse serves a different purpose that follows from operation. Data warehouses store and consolidate the data across all sources to serve a unified analytical or operational purpose whether to provide decision support, customer segmentation, or performance analysis.

### **1.3.2 The scale of operation**

In theory, a database is capable of performing all the tasks of a data warehouse but in practicality this becomes highly inefficient and computationally costly as the data grow larger because both reading and writing operations are much slower because of the relational nature of the database and the need to execute multiple tables joins. Also, a database is usually dedicated to a single application but a data warehouse can consolidate several databases each of them serving a different application.

### **1.3.3 The design fundamentals**

The most fundamental difference is that databases hold records of transactions or operations while data warehouses hold data about a subject. For example, for an ecommerce organization, the database will contain detailed records about the transactions done by the users while the data warehouse will focus on a subject like sales and hold records in fact tables with different granularity (eg. Sales per hour, Sales per age group, or sales store). Also, the data in a warehouse wouldn't be stored in the third normal form for more efficient querying but in the database the 3<sup>rd</sup> normal form would be used to guarantee data integrity.

## 1.4 Introducing the business

The target company is Translatus s.r.o which is a Czech company based and operating in Prague with offices in Germany, Dubai and China. The company started in 2001 as a pioneer in the online freelancing facilitation with a focus on a niche business sector, translation. The company operates under two main web applications.

The first application is a bidding system for translation services along with a project workflow system that allows the customers, translators to manage the project from start to finish. The business model is to get companies or individuals, willing to get translations for their documents, business advertising or any other type of content, to upload their files and then allow translators to offer their services in a competitive manner. Transactions for the purpose are on many sides: bidding, quotes, payment and as part of the design practice we will choose the subject of the data warehouse to be only payment or the final transaction on different dimensions like language pairs, months, translators and clients.

The second application is a community application to write reviews on translators based on previous experiences, similar to Yelp but specific to translators. The translator's is rated in general and according to four criteria: quality, price, timeliness and communication. Suspicious or malicious reviews are verified by contacting the reviewer and asking for more details about the work done with the translator to verify the review is real. Reviewers are free to detail experiences of working with a certain translator, as long as it is kept professional and about the work done without getting into personal rants. In certain cases, the translator would be contacted for more details about the work referenced in the review. The reason to do this is to guarantee the authenticity of these reviews and that they are reliable enough to be the base for future dealings with the translator.

## 2. Thesis Objective and methodology

### 2.1 Objective

The objective of this thesis is to provide a comparison between Apache Hadoop and Microsoft SQL in the perspective of building a data warehouse and evaluating the two solutions through the application of a set of metrics (Butler, et al., 2002). Although, the implementations will be based on a specific example, the metrics chosen are generic to allow for a general evaluation of the two technology infrastructures. More specifically, the metrics chosen are

- **Capacity:** Data warehouse are intended to host all data and more. In transactional databases it is usually the case that we ignore the history of changes in the most tables to avoid adding more layers, relationships and joins. In the data warehouse we want to record everything for future purposes so the intention is to host large and ever growing amounts of data. Capacity, as in the limits for physical storage, is a preliminary measure because if the implementation wouldn't be able to support real life capacity it would be automatically ruled out in practice
- **Loading and indexing performance:** Loading data is the first step in application after the system design is complete and logical design of the data warehouse is implemented on Microsoft SQL Server or on Apache Hadoop. It is a key metric since it defines the usability of the whole implementation because it is unreasonable to use a system that is notably slow at this stage. Even though this an important metric for the system, if a system will be more efficient after this step it could still be favorable for other features since this step is usually done once in the lifetime of the system
- **Client/Server connectivity:** This metric answers the question of future integrability into a complete business intelligence eco system. This is important because it defines the usability and flexibility of using the system. It is important to understand that building the data warehouse is not the purpose; it is just a step in the way of building a



complete data intelligence ecosystem that provides value to the business and as mentioned earlier, one of the main advantages of the data warehouse design is that it facilitates analysis. As such, it is important that the underlying infrastructure that holds the data be flexible enough to allow the use of a diversity of tools by default be it open source or proprietary for the same vendor or for other vendors.

- **Query processing performance:** Now that we have measured the quality of the system in loading the data of the capacity that would be suitable for our needs and made sure that we will be able to use the data warehouse and connect to it from a variety of tools and analysis clients. This metric is what the end users will experience on a daily basis and this defines the efficiency of the system in analytical and reporting uses. For the breadth of the uses that satisfy the business needs, the data warehouse would be responsive enough.

## 2.2 Methodology

The study will also illustrate the different steps in the data warehouse design and implementation, staging, running and utilization for each of the two technologies. The study will begin by a literature review for the main subjects that were encountered in the planning, design and implementation of the data warehouse. Then, the study moves identifying the business requirements and formulating the end product as defined by the business. Next step is analyzing the data sources for the data intended for hosting in the data warehouse. After that, the study shows building the conceptual data model all the way to a physical data model for the data warehouse. Then the study will dedicate separate sections for the actual implementation of the data warehouse in each of the technologies to highlight the differences in implementation, ETL and utilization. The final step would be comparing the two data warehouses and offering recommendations based on the aforementioned metrics

### 3. Literature Review

Database design is one of the oldest and most stable paradigms of computer science. Concepts like normalization and entity relations have been carved in stone. However, the same cannot be said about data warehouses and Hadoop since they are, compared to database design, are new concepts with Hadoop Project being just 10 years old but it wasn't widely adopted until a couple of years later when Google announced that it would sponsor the project through the Apache foundation. This literature review will explore the current state of the art in the two subjects which will help plan and implement the data warehouse in the two technologies.

#### 3.1 Data Warehousing

The literature on data warehousing is in slightly more abundance than it is the case for Hadoop and it is for an obvious reason. While being a byproduct of databases, data warehousing received more attention early. Some date the concept back to the sixties but the first real product to offer the data warehouse architecture was introduced by Devlin and Murphy at IBM in 1988 (Devlin, et al., 1988). This article provided the closest vision to a data warehouse as is perceived now. Those two paragraphs from the article echoes to a great extent the view that is adopted in this project

“The transaction-processing environment in which companies maintain their operational databases was the original target for computerization and is now well understood. On the other hand, access to company information on a large scale by an end user for reporting and data analysis is relatively new. Within IBM, the computerization of informational systems is progressing, driven by business needs and by the availability of improved tools for accessing the company data.”

This excerpt shows the motive for developing data warehouses. With transactional data piling up, the reporting and analysis functionalities were suffering from all the relational complexities put on by the normalization rules. Also, the need for consolidated data reporting was harder to achieve across multiple data sources as the authors specifically made a note of that being the case for IBM itself at the time

“It is now apparent that architecture is needed to draw together the various strands of informational system activity within the company. IBM Europe, Middle East, and Africa (E/ME/A) has adopted an architecture called the E/ME/A Business Information System (EBIS) architecture as the strategic direction for informational systems. EBIS proposes an integrated warehouse of company data based firmly in the relational database environment. End-user access to this warehouse is simplified by a consistent set of tools provided by an end-user interface and supported by a business data directory that describes the information available in user terms.”

The definition of a data warehouse has been subject to minor disagreement between Bill Inmon and Ralph Kimball, two of the most prominent experts on the topic. The disagreement is not about the core functionality of the data warehouse but rather about the scope of its perimeter. In his famous book *The Data Warehouse Lifecycle Toolkit* (Kimball, et al., 2013), Ralph Kimball specifically define a data warehouse as follows:

"The queryable source of data in the enterprise. The data warehouse is nothing more than the union of all the constituent data marts. A data warehouse is fed from the data staging area. The data warehouse manager is responsible both for the data warehouse and the data staging area."

The second definition comes from Bill Inmon, *Building the Data Warehouse*, 1992 (Inmon, 1992)

"A data warehouse is a collection of data in support of management's decision-making process that is subject-oriented; integrated; time-variant; and nonvolatile"

The difference between the two definitions is that Kimball believes that the data warehouse should contain all the business intelligence endeavors and his justification is that since data is the basis for all analysis then when designing the data system; it should be handled with a holistic approach. On the other hand, Inmon looks at the data warehouse in a very abstract way, as a data consolidation utility, without giving much thought and consideration about how the enterprise may decide to the data later on. There are merits to both opinions but the

comparison between the two schools of thought is out of the scope of the literature review, given that both of them agree on the design fundamentals.

### **3.1.1 Entity-Relationship Modeling**

Entity-Relationship model is an application design technique that is used in transactional databases. The technique is based on the idea of not repeating data and keeping only one source of truth for every entity, a table or master table, and then references it using “foreign keys” in other master tables that need to reference it. It is beneficial in the sense that it enforces strict rules on inserting new records that ensures data integrity and that it is also a space efficient since the entities’ data are stored only once. However, the relational model itself has its own shortcomings. The biggest shortcoming of all is that the database could easily become very complex and deep with several entities all referencing each other.

If complexity on and of itself, wouldn’t be considered a drawback, consider the effect of this complexity on performance for large queries. Joins are considered to be the most complicated and computationally demanding part of the relational data queries (Mishra, et al., 1992). The second major disadvantage is the design inherent inability to record the history of changes in the entities’ information. For example, if we have a master table “Customer” that has an attribute “Address” if later the customer address changed the old address isn’t saved historically. The only solution would be to add another relationship layer and separate address as another entity and thus making the design ever more complex. The following figure illustrates this needed transformation

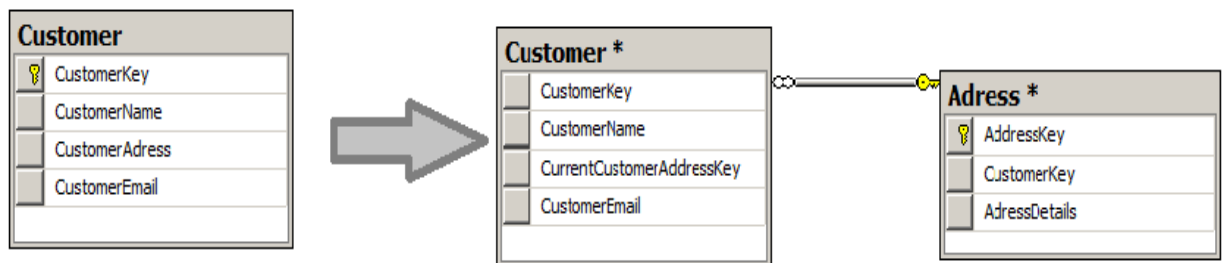


Figure 2 Type 1- transformation [source: own]

After this transformation it would be possible to retrieve previous addresses from the address table using **CustomerKey** and the **AddressDetails** of the current address through **CurrentCustomerAddressKey**

Another problem of Entity-Relationship is that the design of the database becomes unreadable and hard to understand or explain which later makes the process of designing complex queries hard and error prone.

Another major disadvantage of using Entity-Relationship is that it is closer to the programming world but rather far from the business itself (Kimball, et al., 2013). It models the relationships between the data elements but not the context of these relations in the business model.

Lastly, the entity relationship model is difficult to modify since you need to break the integrity rules and reassemble new rules and relationships or add a new relationship layer and make it more complicated. For example, the previous transformation for the address would require first to create the **Address** table with the foreign key to **Customer** table. Second, store the address details from **Customer** table to **Address** table. Third, remove **CustomerAddress** field from **Customer** table. Fourth, create a new column in **Customer**, **CurrentCustomerAddressKey**. Lastly, fill **CurrentCustomerAddressKey** with the current addresses from table **Address** using **CustomerKey**.

### 3.1.2 Dimensional Modeling

Dimensional modeling is the common terminology for the data warehouse logical design technique. The term was coined by the Metaphor group (Kimball, et al., 2013), as opposed to the Entity-Relationship modeling technique. This paradigm aims to eliminate the drawbacks of the entity-relationship modeling to provide more efficient, readable and flexible alternative. The key idea is simple, to keep the design flat. The dimensional model consists of a **fact table** that has foreign keys to multiple **dimension tables**.

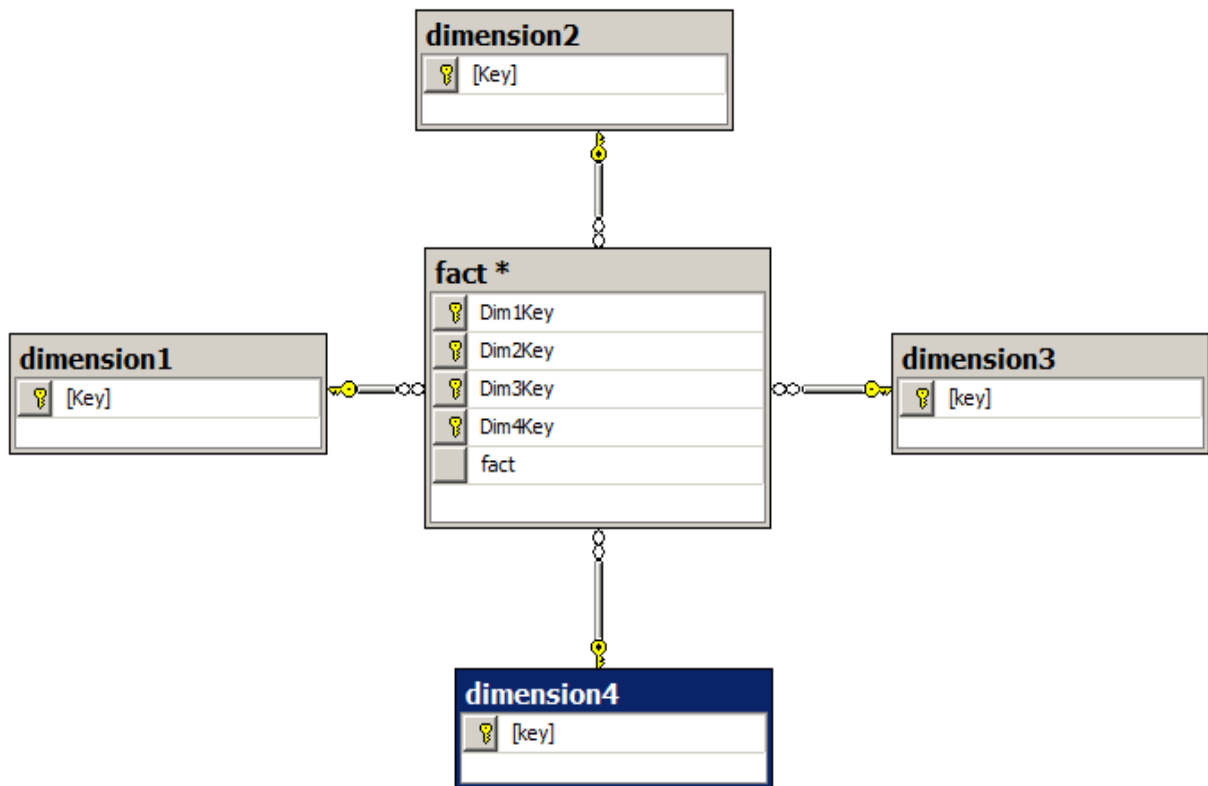


Figure 3 Dimension Model example [source: own]

### *3.1.2.1 Fact Table*

Fact tables are the main heroes of the data warehouse and choosing them is usually the first step in creating the dimension model; they contain the data that we are actually interested in for analysis; sales, working hours, stock levels. In essence, we only need the fact table to describe and analyze our data with minimal or no need for dimensional tables (Devlin, 1996). Fact tables typically consist of two or more foreign keys to dimension tables and one or more **facts**.

A good practice is that the **facts** will be additive or can be aggregated. For example, Sales per Day as a fact described by foreign keys to product table or time tables. At the time of choosing the facts that the design will revolve around, based on business needs of course, an important decision regarding fact tables is to define the granularity of the fact table. In this context, granularity is the aggregation measuring unit of the fact item in the table. For example, are we interested in saving the sales per hour, per minute, or per day? Making this decision is crucial for the next step in dimensional modeling, choosing the dimensions.

### *3.1.2.2 Dimension Table*

A dimension table is used to describe one aspect about the data in the fact table. After choosing the fact tables and their granularity, comes the time for designing the dimensions that will describe the fact tables. While the dimension tables could be ignored for the analysis stage, they are crucial for reporting and understanding the data in the final presentation. Dimension tables are descriptive by nature and discreet and by acting as a single source of truth for the dimension, the dimension tables should be complete and accurate to avoid errors in the analysis stages and simplify querying.

Every row in a dimension table must describe only item. There exist 3 types of dimension tables based on how the design handles change in the data aspect represented by the dimension table (Devlin, 1996). **Type-One** dimension table is when the design decision for this dimension is to ignore the history of change and only maintain the latest information

about the dimension. **Type-Two** dimension is when the design decision is to maintain history for the items in the dimension by adding a new row that contains the new information and specify in the appropriate columns the expiry of the old data and the activation of the new data for this item. The following table (Table 1 Type-Two Dimension Change [source: own]) illustrates an example of a type-two change

**Table 1 Type-Two Dimension Change [source: own]**

<b>RowKey</b>	<b>RowUid</b>	<b>RowInformation</b>	<b>ActiveDate</b>	<b>InactiveDate</b>
<b>1</b>	1	Old Info	1/1/2001	20/2/2016
<b>2</b>	1	New Info	20/2/2016	31/12/9999

Important things to note from this example are that first the RowUID identifies the item and RowKey Identifies the data row itself in order not to disturb old analysis when we add a new record and second the ActiveDate & InactiveDate which are called the activation columns.

By default the end of the old row validity is equal, or in some systems, one time unit less the new information active date. Last type of dimension tables is **Type-Three**, in this type when the changes affect the whole table and the design decision is to add a new column to preserve the history and reduce the redundant data. The following table ( Table 2 Type-Three Dimension Change [source: own]) reflects the previous example in the case of type-three change

**Table 2 Type-Three Dimension Change [source: own]**

<b>RowKey</b>	<b>RowUid</b>	<b>RowInfo</b>	<b>NewInfo</b>	<b>ActiveDate</b>	<b>InactiveDate</b>
<b>1</b>	1	Info	New Info	1/1/2001	31/12/9999

Choosing and designing the dimensions follows from the decisions taken regarding the fact tables, especially granularity.



### *3.1.2.3 Why Dimensional Modeling?*

There are many reasons to confirm that choosing the dimensional model over the entity relationship model for the data warehouse design is the correct choice. The dimensional model excels in the areas where that the entity- relationship model fell short. First, the dimensional model is a comprehensible, standard framework. Writing reports, using query tools, and designing user interfaces can all become based on strong assumptions about the dimensional model to reflect the business interest, and to provide efficient processing. For example, since the dimension tables are defined based on the business preferences it becomes very easy to browse through these dimensions or read them out from the reports without the need for double or triple joins to get to the correct sub-dimension or sub-sub-dimension.

Not only that, this sense of predictability offers great gains in processing.

It is common for business intelligence analysts to use query optimizers which are usually based on query costs and rather ignores the semantic meaning of the underlying tables. With dimensional modeling, on the other hand, the database engine would fixate the dimension table and match the values using a Cartesian product operation based on the keys (Join Processing in Relational Databases, 1992). This makes it possible to integrate all the dimension table data in a single pass which offers a huge performance boost to the end user and decreases the need for more computational resources. In short this predictable nature benefits both the server and the analysts and that leads to a better presentation and performance.

A second advantage for the dimensional model is that the star join schema offers more ease at dealing with requirements change. The conformed dimensions stay the same and could be reused in new facts if required. The dimensional model is that it is naturally extensible and able to accommodate unexpected new data requirements and changes in design decisions. This is basically possible because the dimensions are not a subject to change so if the need arises to change the perspective of the fact table then based on that change it either go down to a basic **ALTER TABLE** SQL command in place to add a column or remove another, an **UPDATE**

statement to change the granularity or by adding a new fact table with different aggregation or granularity. It is important to note that depending on the change a data reload might be required. For example, if the current granularity of a fact table with regards to a time dimension is monthly and the decision is to switch back to a weekly or daily granularity. However, in all cases, if the dimension model was correct from to begin with, it shouldn't be required to change the structure of the tables or break the foreign keys as it is the case with entity-relationship model. Also, the underlying applications or reports wouldn't be affected by the change. Here are some common changes that occur during the data warehouse development process:

- Adding new unanticipated facts to an existing fact table of the same granularity which translate into an ALTER TABLE statement to add the new fact as column and then create the ETL procedures required to fill it.
- Adding a new dimension, sometimes a new dimension emerges from observing repetitions in fact tables and it is a good choice to consolidate it into a separate dimension to increase data integrity and speed up reporting
- Adding new attributes to existing dimensions.

A third advantage for the dimensional model is that it uses the same language as business analysts to represent the business data which results in better communication regarding the implementation and the challenges along the way. Dimensional modeling also borrows the same solutions to some of the common modeling problems like slowly changing dimensions, heterogeneous product dimension, or many other business modeling challenges.

## 3.2 Apache Hadoop

Apache Hadoop is an open source project sponsored and maintained by the Apache organization. The project is a framework for distributed processing of large data sets using clusters of computers while providing a simple programming models and managing interface (White, 2015). The design of the library has been tailored for high-availability in mind in order to reduce the need for hardware redundancy. The library detects and handles failure at the application layer allowing delivering a highly-available service using a cluster of machines each of which is prone to failure. The framework is designed in a modular way to minimize the overhead of unnecessary functionality or seamless addition of needed functionality later on. The modules are tied together using another module, Hadoop Common. Hadoop Common provides the common utilities that support other modules. It acts as the core of the framework managing the interaction with the underlying operating system and file system as well as the starting and shutting down operations.

The second module is Hadoop Distributed File System or HDFS. HDFS is the master piece in the framework making it possible to process files in a distributed manner in the map and reduce operations. The HDFS looks at the cluster as a whole rather than per cluster; data is divided into smaller pieces called blocks and blocks are mapped throughout the cluster making it possible to process smaller subsets of data on each cluster and thus utilizing the full power of the cluster (White, 2015).

The third module is Hadoop YARN, short for Yet Another Resource Negotiator, a job scheduling, application management and cluster resource management framework. YARN itself is split into two main components, Resource Manager, Node Manager and Application Master. The Resource Manager is the main authority over all the resources of the Hadoop cluster and it controls the interaction with the clients. Node Managers are similar to the Resource Manager but are limited in their scope of authority to a single node and they don't interact with clients and rather report the required metrics and resources usage to the Resource Manager. Application Masters are resource negotiators for individual applications to handle scheduling, monitoring and progress of these applications. The following figure illustrates the inner architecture of YARN

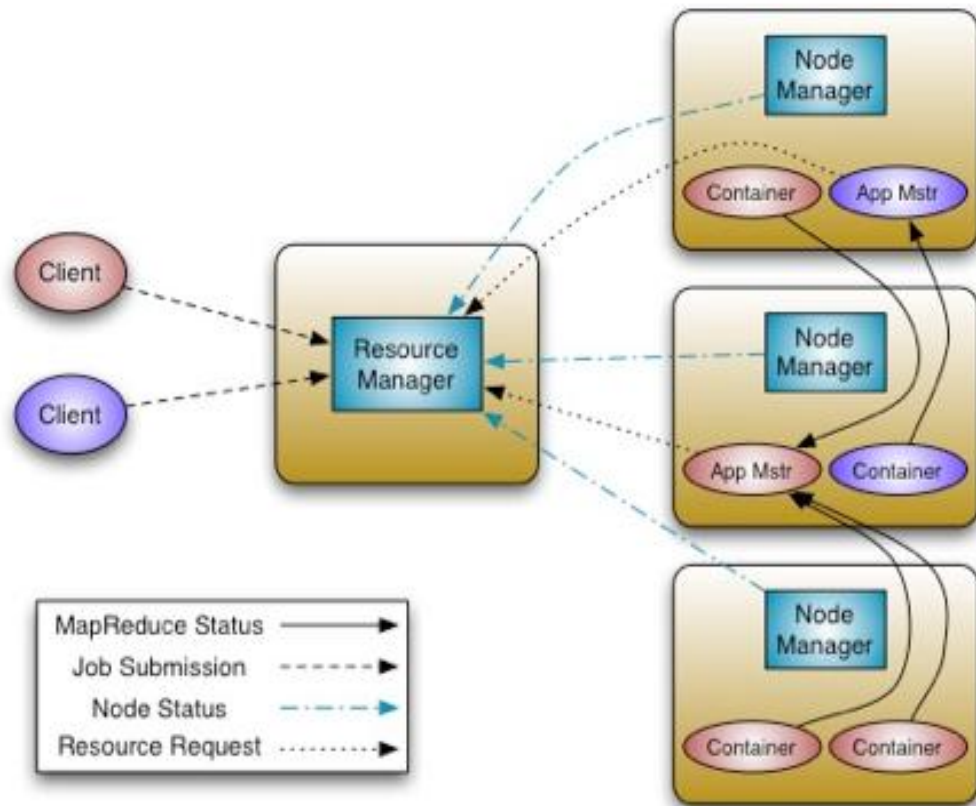


Figure 4 Hadoop underlying system architecture [source: (Apache Software Foundation, 2016)]

So far Hadoop has not been used in practice as a data warehouse infrastructure. However, it has become an industry wide advice to use Hadoop to stage the data before feeding it to the data warehouse. One reason for this is that Hadoop is cheaper and was built for efficiency in raw data transformation through the MapReduce processes.

### 3.2.1 YARN

YARN as a resource negotiator borrows some ideas from the common architecture of a Server-Client. A global ResourceManager instance in this case acts as the server element and makes decisions about dividing available resources on the requesting applications (Grover, et al., 2015). To do this, the ResourceManager must track the nodes in the cluster and request information about the available resources and whether the node is still available or not.

After obtaining the information about the resources the ResourceManager bears the responsibility of making the allocation or scheduling decisions. The decision making process is not trivial because of the multiple factors that are considered with each decision like priority for the requesting application, maintaining data locality within the cluster if possible, access levels and probably other user defined factors.

The second part of YARN is the NodeManager, which is very similar to the ResourceManager but with a scope limited to a single node in the cluster. The NodeManager is also responsible for reporting the status of the node to ResourceManger including resources and a periodic pulse to confirm that the node is still available. The NodeManager creates resource containers to manage and contain the information and processes utilized by every application like CPU usage, memory usage and disk and network IO. The number of these resource containers is dynamic and can be changed from the configuration but the resources of the node should be considered when setting the number to avoid having the NodeManager queuing many resource containers that each of them shares a very limited amount of CPU or memory resources.

For every application request submitted to the ResourceManager, a smaller process is created ApplicationMaster. This ApplicationMaster process is responsible for managing the execution of this request and offer many utilities for the user like automatic restart of failed requests and monitoring. The ApplicationMaster and its process are contained in a resource container in the NodeManager. The ApplicationMaster can run any type of task inside the

resource container and even custom ApplicationMaster could be created following the Apache specification. This is particularly useful because it makes YARN extensible for any application as long as it implements an ApplicationMaster that follows the specification. This is usually helpful with new data formats where utilizing the data itself doesn't follow the normal IO process like an application that would contrast X-Ray images. The ResourceManager is responsible for determining which ApplicationMaster would be initialized for every application request similar to how any operating system decides which application to open a certain filetype. The specification for the ApplicationMaster is also simple and concise (Holmes, 2014). It must send a resource request to the ResourceManager in this request the following should be specified:

- Amount of resource like memory, disk in megabytes and CPU.
- The preferred node to maintain data locality if possible and reduce network traffic or '\*' if there is no specific preference
- Priority for this request in light of other requests for the same application

The resource manager replies to this request by a container that satisfies the request details, specified by ID and node name, and then the ApplicationMaster will ask the NodeManager on this node to use this container to start the execution. The monitoring is provided by the ApplicationMaster, however, the NodeManager can kill the container if it exceeded the requested application.

The short description of the three components of YARN is that they are abstractly the same and act as resource managers but with different scope; ResourceManager has the full scope of the cluster; NodeManager has the scope of a single node; ApplicationMaster has the scope of a single application request.

### 3.2.2 Hadoop Distributed File System

Hadoop Distributed File System (HDFS) is the core of the Hadoop environment and is the core that enables Hadoop to be efficient, reliable, flexible and cheap. In a Hadoop cluster, the data is divided into small blocks and then these blocks are distributed across the cluster. When the time comes for processing the data, Hadoop, specifically the MapReduce processes, are able to process each of these blocks simultaneously and easily scaling computing across the cluster. HDFS was built to automatically handle fault in the cluster. For example, if we have a cluster of 100 servers and each server has 4 internal drives

HDFS will distribute the data so that the small blocks are replicated across the cluster while automatically keep track of replicas and the status of each server through the Unit Managers and Resource Managers. The level of redundancy can be modified on all levels; file, server, cluster. However, the default behavior of Hadoop is that it would replicate each of the blocks on two other servers for failure handling (Grover, et al., 2015). However, HDFS handles replication in an extremely smart manner that makes it different from all other file systems.

HDFS is rack and network aware and uses this information to handle data replication such that it would cause minimal network usage or disturbance and with consideration to the receiving node operations at the time of replication. The term **block** has been used to describe the division of files in HDFS so it is important to put some emphasis on this division process. One of the core goals for Hadoop is to host extremely large files and provide the means to use these files and extract information from them. In doing this, HDFS divide each file to blocks, of configurable size. The default size of a block is 64 megabytes (White, 2015).

Once the user initiates the process of creating or adding a new file on, HDFS accumulates data from the source until it receives enough to fill a block, a block is created and the block identifier is passed to the name node. The same process continues till all the data has been received and assigned to blocks. The name node then takes each block writes them to disk and decides where the replica would be placed and initiates the replication process. HDFS also manages many other tasks that in other systems would be handled manually and would

require written procedures and a lot of effort. Data rebalancing is the first of these tasks. Through the life time of the cluster it is highly probable that some sort of imbalance would happen leaving some nodes with more data than others because of how the replication process was handled or simply because the node has been upgraded with extra memory.

Data integrity maintenance is a key feature for any storage system and again HDFS manages this part of the system. During the block creation process HDFS would create checksums for each block and store these checksums. Later the name node would use these checksums to verify the integrity of the blocks and would automatically replace the defected blocks using their replicas or report them to the ResourceManager.



### 3.2.3 Hive

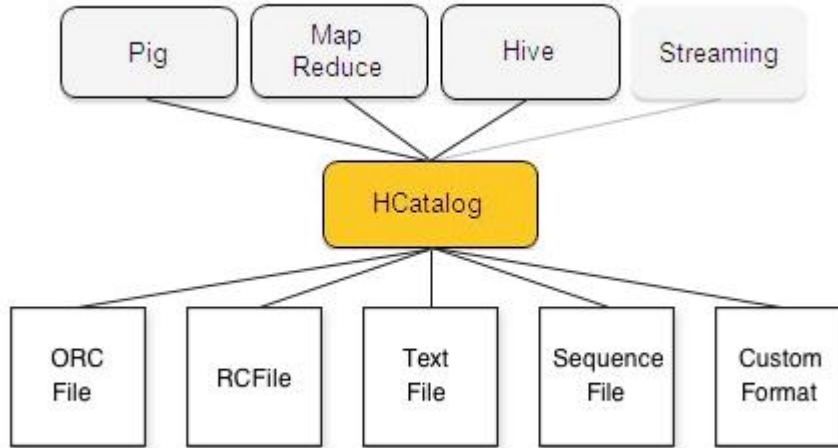


Figure 5: HCatalog in Hadoop [source: (Leverenz, 2016) ]

Another important part of the Hadoop environment is Hive. Hive is the meta-data manager for Hadoop; it adds another layer between HDFS and the querying tools to define the data. The equivalent to Hive in the normal Microsoft SQL based systems is the schema layer. However, unlike Microsoft SQL, Hive schema, which is managed by a micro layer called hCatalog, is not actually bound to the data. So at any point of time the whole catalog could be removed or edited without any need to change the data (White, 2015). Also, most of the time, in the Hadoop environment, creating the catalog is done after adding the data. Defining the metadata and table-like information is a considerable amount of work but there are many benefits that come with it. By creating a middle layer, the interaction with the data becomes much easier and standardized which means the ability to use more querying tools like Pig and MapReduce(Leverenz, 2016). hCatalog provides a log for what other users have created and enable sharing the work and the results as well which is important for big teams. Finally, hCatalog provides a REST API which means that it could be integrated within other enterprise systems already in place and benefit from the role based access level ...etc.

After defining this meta-data definition layer, Hive makes it possible to query the data using SQL-Syntax. Hive then takes the SQL and converts it to a series of auto-generated MapReduce programs to optimize the performance to run on the Hadoop cluster

### 3.2.4 Why Hadoop?

So why are we bother studying Hadoop as an alternative when there are so many stable alternatives from technology giants like Microsoft, Oracle and IBM for building data warehouse. They are even offered with the hardware or cloud hosting if it suits the user. The answer to this question is really simple; Hadoop is cheaper, more distributable, and more flexible.

#### 3.2.4.1 Cheaper

The main components of any information system are software and hardware and it is not any different for building a data warehouse. Hadoop is open source which means that the user would not have to pay huge licensing fee and it runs on Linux which is also open source. Also, Hadoop is hardware agnostic which means the user wouldn't have to invest in new hardware to start deploying Hadoop. Hadoop handles cluster failures at Application level and thus eliminates the need for high availability and redundancy hardware. A more subtle point is that Hadoop applies a concept known as Data Locality which implies that the data will be processed on the same machine where it is stored, when possible, and thus decreasing the total network overload significantly and eliminating, or even discouraging, the idea of using network attached storage (NAS), or storage area network (SAN). The following is sample cost comparison for 3 years for an Oracle Data Warehouse and a Hadoop Cluster that would host 300 terabytes of user data. (Oracle, 2016)

Table 3 Costs of Oracle Data Warehouse [source: own]

	Year 1	Year 2	Year 3	3-year total
Hardware	\$ 525,000			
Annual Support Cost	\$ 63,000	\$ 63,000	\$ 63,000	
On Site installation Costs	\$ 14,000			
<b>Total</b>	<b>\$ 602,000</b>	<b>\$ 63,000</b>	<b>\$ 63,000</b>	<b>\$ 728,000</b>

Table 4 Costs of Hadoop Data Warehouse [source: own]

	Year 1	Year 2	Year 3	3-year total
Hardware & Networking	\$ 397,000			
Annual Support Cost	\$ 55,000	\$ 55,000	\$ 55,000	
On Site installation Costs	\$ 15,000			
<b>Total</b>	<b>\$ 467,000</b>	<b>\$ 55,000</b>	<b>\$ 55,000</b>	<b>\$ 577,000</b>

The two tables illustrate that the cost of Hadoop is 20% less than an Oracle. However, these numbers doesn't illustrate the fact that the cost of adding extra storage capacity to both data warehouses, Hadoop would be 50 times cheaper for the cost of adding 1 Terabyte.

#### 3.2.4.2 Distributable

Hadoop can scale from a single machine to thousands and when adding new servers to the clusters there is no need to change the original setup or reload the data. Hadoop will

automatically reallocate the resources making every unit an independent computing and storage unit of its own. If the need arises to enhance the cluster by adding one or more new servers, the new server is added to the cluster then Hadoop will start by allocating blocks of data and then declaring readiness for processing to the master Resource Manager

#### *3.2.4.3 Flexible*

Hadoop doesn't care about the nature of the data or file formats. Data files of all forms could be stored and later used for analytics provided the user creates the proper MapReduce procedures for these formats. Hadoop was originally designed just to store data in clusters. In many cases, the need to store the data in its original format is crucial to the business need. For example, medical data, X-rays, doctor's notes, data gathered from sensors or other IoT device are usually better being saved unaltered. And then using MapReduce we can develop the queries that extract the pieces of information that are needed when they are needed without any loss of information or needing a secondary storage to store the raw data and later reloading the raw data with the new extraction and transformation rules.

Not only that, Hadoop doesn't care about the underlying hardware or operating system, which would allow the user to have as many operating systems, hardware architectures in the cluster without any extra effort for configuration. Hadoop creates an abstraction layer that only cares about the connectivity between the nodes and all the management would be then done using the NodeManagers and ResourceManagers

## **4. Project Implementation**

### **4.1 Business Requirements**

As with any project, business requirements not just guide the implementation but in essence create the need for the project as a whole. Gathering the requirements passed through many phases to reach a proper level of specificity that guarantees accurate implementation and meeting the actual business needs. First, general objectives were gathered from the management to identify what are the key areas they expect enhancement as a result of investing in the data warehouse. The main purpose behind developing the warehouse from the business management point of view is to remove the limitations on reporting and enhance planning through predictive reporting. As with all the first tiers the requirements were broad and included a full vision for the whole project but not specific enough to put in motion a first stage of implementation. As more and more refinement rounds and interviews were carried on the following requirements were reached.

The process enabled the division of the requirements to Information requirements, analytical requirements and technical requirements.

#### **4.1.1 Information Requirements**

Information requirements will guide the implementation on what data the data warehouse needs to store to satisfy the business intelligence objectives. Also, collecting the information requirements will be the basis at the later step of analyzing the current data sources and be able to skim through to extract the most useful table and relations from these data sources.

- Client information  
Client line of work, documents domain, recurrence, seasonality, documents formats submitted and volume.
- Translator information  
Language support, responsiveness, rating, availability, pricing and reliability

- Sales information  
3 years of sales history, profitability (quote compared to translation fees,  
languages and document formats delivered)

#### 4.1.2 Analytical Requirements

Analytical requirements define the business intelligence needs and while being out of our main interest but it can provide useful directions regarding the design decisions and choosing fact tables and granularity. The sales division and management expect the warehouse to provide better reporting capabilities that would help them understand the existing customers' requirements and have a way to predict and plan the work requested. They also want a better and more pragmatic method to choose translators based on previous work delivered or in some cases not delivered. A better and more concise formulation for these requirements is summarized in the following analytical questions.

1. How much revenue have we generated with a specific customer or group of customers over the last 6 months?
2. How much revenue have we generated for a specific language or group of languages over the last 6 months?
3. How much revenue have we generated for a specific domain or a group of domains over the last 6 months?
4. How much revenue have we generated by a translator or a group of translators over the last 6 months?

5. How much profit have we made from a translator or a group of translators over the last 6 months?
6. Do we need more translators for a certain language based on the number of sales done for this language?

#### 4.1.3 Technical Requirements

Technical requirements are the business directions for the implementation specifics in case the company has some restrictions on hardware or software choices. The company's current infrastructure is Microsoft oriented with business licenses to Windows7, Windows Server 2012, SQL Server 2008 and Visual Studio 2013. However, it is important to note that for the purpose of this study we will note that the company is neutral for the new data warehouse project as long as the implementation satisfies the technical and analytical needs. The volume of data for the aforementioned application databases is estimated to be 700 gigabytes. The hardware resources that would be dedicated to the data warehouse project in the first stage are:

- 2 x 2.6 GHz Xeon E5-2650 v2 processor (1 chip, 8 cores) with 20 MB L3 cache
- 8 x 8GB (1 x 8GB) Single Rank x8 PC3L-12800R (DDR3-1600) Registered CAS-11 Low Voltage Memory Kit
- 4 x 4TB 6Gb SAS 7.2K LFF hot-plug SmartDrive SC Midline disk drive (3.5")
- 1 x ProLiant DL380p Gen8 Rackmount, 8 SFF CTO Model (2U) with no processor, 24 DIMM, open bay (diskless) with 8 SFF drive cage, Smart Array P420i controller with Zero Memory, 3 x PCIe 3.0 slots, 1 FlexibleLOM connector, 4 x redundant fans, Integrated HP iLO Management Engine
- 1 x HP 1GbE 4-port 331FLR Adapter
- 460W Common Slot Gold Hot Plug Power Supply

## 4.2 Analyzing Data Sources

After defining the technical requirements in their final form, the next task, naturally, is to define to analyze the current data sources. This step will be the foundation for dimensional modeling and designing the ETL processes. The aforementioned company currently owns two applications TD Apro.com & TranslatorAdvisor.com. TD Apro.com is a translation service facilitation website that aims to act as middleman between translators and customers seeking translations for various materials. From the interviews, it is clear that a great deal of offline communications and operations still happens and is unaccounted for in any of the databases but that wouldn't affect the realization of the data warehouse objectives.

TranslatorAdvisor.com is a rating and feedback service for translators, similar to Yelp.com. Based on the business and technical requirements, some aspects of both databases will be consolidated in the data warehouse. The consolidation required is in the areas related to the translator ranking based on previous experiences.

### 4.2.1 TD Apro

The TD Apro database is built for project management and workflow, bidding and sales transactions as well as capturing communications between buyers, translators and project managers. Based on our information and analytical requirements, it becomes clear that the focus is on the final sales transactions and the translator evaluations and we can safely ignore the segments of the database that relates to project management and bidding and only focus on the sales part which also includes the information about the clients and translators involved in each transaction. In the figures below is an overview of the database in general highlighting the segments that would be used to realize the objectives of the data warehouse. The following figure illustrates the parts of the TD Apro application database that we will incorporate in the data warehouse.



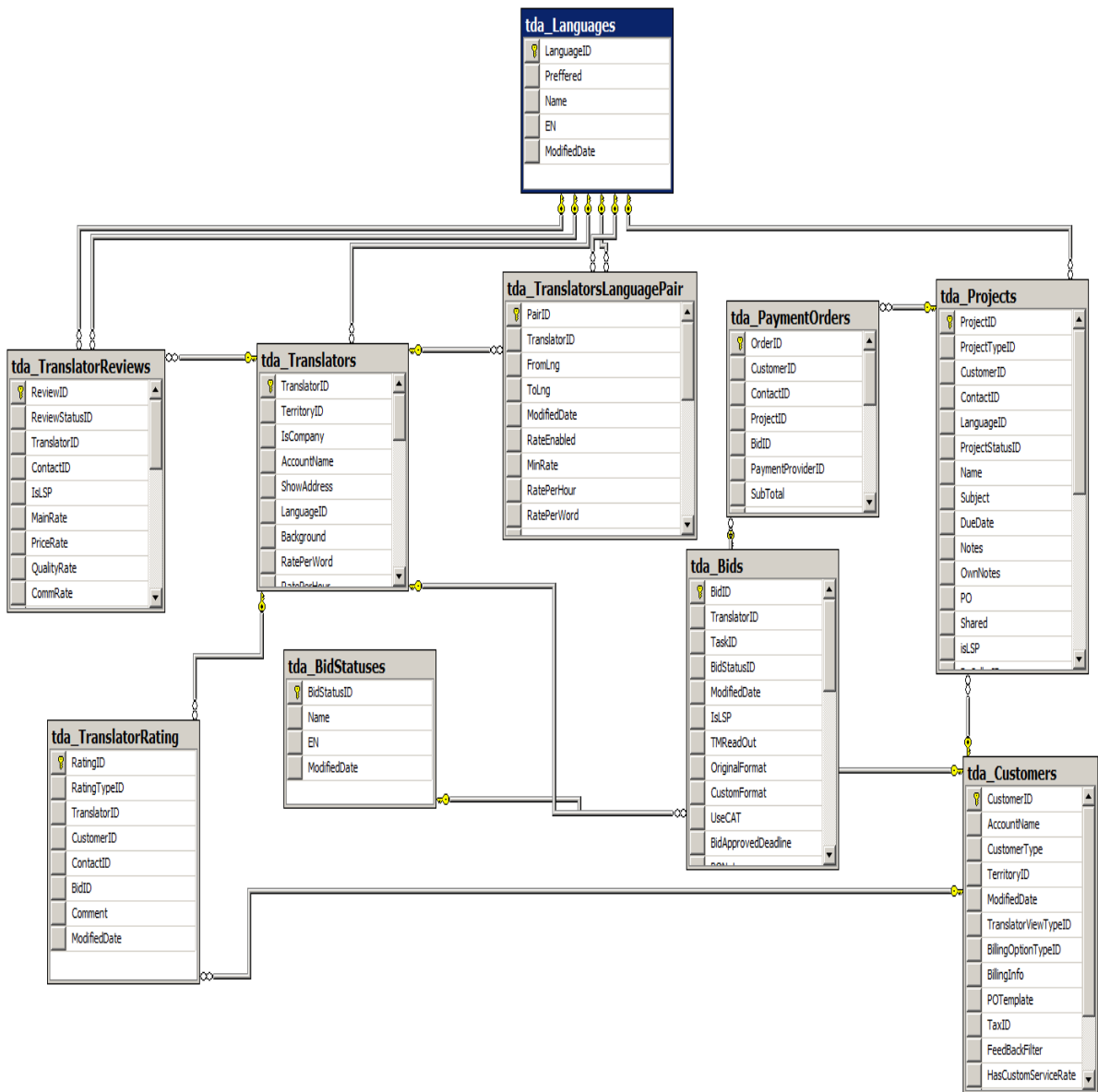


Figure 5: Simplified TDApro application database [source: own]

The key things to look for while analyzing the data source are

- The deep relationships that would need to be flattened for the dimensional modeling process

- The aggregation conditions, from business and technical perspective, that should be maintained
- The elimination cases that would be used to eliminate irrelevant and redundant data

Following these three points the first issue was with the LanguagePairs table which in the entity relationship model extends itself to another table for a Many-to-Many relationship with the translator table and the table itself is a Many-to-Many relationship with Languages table. To flatten this part of the model the language table will be kept in place but the LanguagePairs and TranslatorLanguagePairs would be eliminated and replaced by six columns in the translator dimension, two for each language pair.

The second decision will be to eliminate all the records that are generated during the negotiation process. For example, for a single successful transaction there could be more than five records in the bids table

#### **4.2.2 TranslatorAdvisor**

TranslatorAdvisor is a customer feedback website. Anyone who had a previous experience with any of the translators is able to give feedback regarding this encounter and rate the translator in various sections such as price, quality, communication and timeliness in delivery. The main difference between TranslatorAdvisor and the feedback feature on TD Apro is that TranslatorAdvisor is not restricted to translators who are registered and agreed to work with the company and the feedback is not related to a certain project and not limited to customers who got translations through TD Apro; it is open for all customers and all translators. For our data warehouse, this means that a decision will have to be made about whether to include all the translators in the warehouse or limit it to those registered on TD Apro. The following figure shows the parts of the TranslatorAdvisor application database that will be consolidated into the warehouse.

## 4.3 Dimensional Modeling

Based on the business requirements we have confirmed and the analysis of the data sources, the first phase of the data warehouse will focus on sales transactions and translators evaluations. The dimensional modeling process itself would be divided into three main steps:

- Designing dimensions
- Designing fact tables

### 4.3.1 Naming Terminology

Choosing a naming terminology is a standard and a useful practice. It makes it even easier to understand the dimensional model and quickens the query writing process in the future. For our data warehouse we chose a simple yet powerful naming scheme. All table names will be Camel Case without spaces or underscores. Dimension tables will start with the prefix **Dim** and fact tables will start with the prefix **Fact**. For example, **DimCustomer** will be the table name for the customer dimension while **FactSales** will be the fact table for sales.

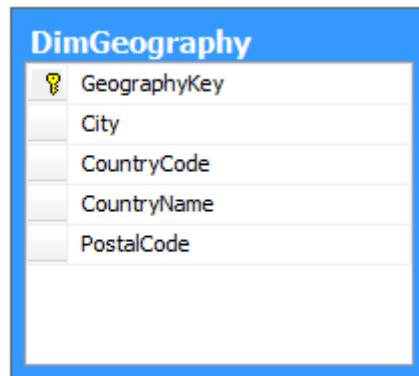
### 4.3.2 Dimensions

Dimensions are the pillars for understanding the model and after designing them, designing the fact tables becomes much easier and less confusing. Many of the design decisions are taken while designing the dimensions so in this section, the design process and the decisions taken will be illustrated.

#### 4.3.2.1 DimGeography

The Geographical distribution of the customers have great business significance for the company because it also defines some aspects about the nature of payment for the translators and the customers and some other legal aspect and also, as it comes to translation, it gives useful information about the local dialect required and the time zone for delivery deadlines from the perspectives of customer, the translator and the project manager. The dimension has the following attributes:

- *GeographyKey*: The primary key for the dimension
- *City*: The name of the city
- *CountryCode*: The iso code for the country
- *CountryName*: The full country name.
- *PostalCode*: the postal code for this geography dimension row




DimGeography	
	GeographyKey
	City
	CountryCode
	CountryName
	PostalCode

Figure 6: Geography Dimension [source: own]

#### 4.3.2.2 DimCustomer

From the name, this is the dimension table for customers' information and it will contain the following information:

- *CustomerKey*: the primary key for the dimension

- *GeographyKey*: A foreign key to the customer geography dimension that stores information that complements other attributes for contact information
- *Name*: The name used for customer contact during operation
- *EmailAddress*: The email used for customer contact during operation
- *Enterprise*: a Boolean indicator for whether the customer is an enterprise or an individual account
- *YearlyIncome*: The last recorder annual revenue for the customer, for Enterprise customers
- *CustomerDomain*: The main domain of interest for the customer gives useful information for the translator and for the analysts
- *AddressLine1* & *AddressLine2*: The address used for recorded billings and offline communication
- *Phone*: the phone number used for communication
- *DateFirstPurchase*: The date of the first successful operation


DimCustomer	
	CustomerKey
	GeographyKey
	CustomerAlternateKey
	Name
	Enterprise
	EmailAddress
	YearlyIncome
	CustomerDomain
	AddressLine1
	AddressLine2
	Phone
	DateFirstPurchase

Figure 7: Customer Dimension [source: own]

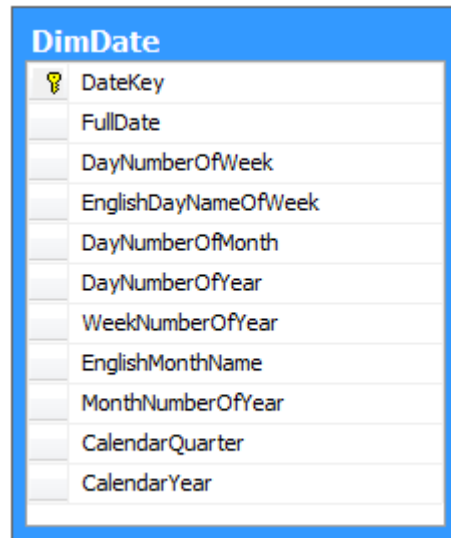
#### 4.3.2.3 DimDate


The date dimension is one of the key parts of the dimension model because it will be interacting with many other parts of the model and definitely with the fact tables as well. The date dimension in our case will also define the granularity of the fact table. The following attributes compose the dimension:

- *DateKey*: the primary key for the dimension table. For this table we will not use the normal or default primary key convention of having a consequence of numbers. Instead, we will compose the value of the primary key to reflect the date itself. In this way we guarantee that we will not have duplicate rows for the same date. Also, if the need to add older dates that already contained in the dimension, there would be no need to do any changes to the indexes because the new dates would automatically be in the correct order. For example the primary key for 31/12/2001 would be 31122001.
- *FullDate*: The date represented by the row (31-12-2001)
- *DayOfTheWeekName*: The English name of the day (Sunday – Monday ...etc)
- *DayNumberOfTheWeek*: The order of the day in the week from one to seven
- *DayNumberOfTheMonth*: The order of the day in the month or the first component of the date
- *DayNumberOfTheYear*: The order of the day in the year from 1 – 365
- *MonthName*: The English name of the month (January, February ..etc)
- *MonthOrder*: The order of the month in the year from 1 – 12 or the second component of the date
- *Quarter*: The quarter that this day lies on from 1 – 4
- *Year*: The year part of the date (2005 – 2006 ..ie)

It could be a bit confusing to many to see attributes like *DayNumberOfTheMonth*, *DayNumberOfTheYear*, *Quarter* Or *Year* because they violate the normalization rules to not include repetitive attributes and separate them in a different table..etc. However, these attributes are required for the analysis and reporting purposes and these information must be included so although they are a bit redundant and repetitive and some of them are

computable, it is a better idea to maintain the flat nature of the dimension model to keep the performance gains of this approach.



DimDate	
	DateKey
	FullDate
	DayNumberOfWeek
	EnglishDayNameOfWeek
	DayNumberOfMonth
	DayNumberOfYear
	WeekNumberOfYear
	EnglishMonthName
	MonthNumberOfYear
	CalendarQuarter
	CalendarYear

•  
Figure 8: Date Dimension[source: own]

#### 4.3.2.4 DimLanguage

The language dimension is another multi-purpose dimension that is part of other dimensions and the main fact table. It will be part of the translator dimension later and the sales fact table. It is key in the analytical needs because it helps set many directions for the management and sales people. This dimension and its incorporation in other dimensions and fact tables will be critical to satisfying many of the analytical needs identified earlier. The dimension will be composed of the following attributes:

- *LanguageKey*: The primary key for the dimension table
- *LanguageCode*: The iso code of the language (en, cs, fr ... etc)

- *LanguageName*: Full language English name (French, Czech, English ...etc)


DimLanguage	
	LanguageKey
	LanguageCode
	LanguageName

Figure 9 Language Dimension [source: own]

•

#### 4.3.2.5 DimTranslator

The second most important actor in the sales transaction and sales fact table as well. This dimension would answer many of the analytical requirements and identify the business need for recruiting more translators in a more intelligent way.

This dimension was especially tricky in design because it belongs to both data sources in a different representation and because it had many levels of join and some decisions had to be made to flatten these data and make the dimension follow out dimensional modeling approach. The following is how the dimension is composed:

- *TranslatorKey*: The primary key for the dimension
- *GeographyKey*: A foreign key to the geography dimension
- *NativeLanguageKey*: A foreign key to the language dimension table to represent the translator's native language
- *FirstLanguagePairFrom*: A foreign key to the language dimension table to represent the "from" part of the first language pair for this translator (ie. **English** -> Czech)
- *FirstLanguagePairTo*: A foreign key to the language dimension table to represent the "to" part of the first language pair for this translator (ie. English -> **Czech**)
- *SecondLanguagePairFrom*: A foreign key to the language dimension table to represent the "from" part of the second language pair for this translator



- *SecondLanguagePairTo*: A foreign key to the language dimension table to represent the “to” part of the second language pair for this translator
- *ThirdLanguagePairFrom*: A foreign key to the language dimension table to represent the “from” part of the third language pair for this translator
- *ThirdLanguagePairTo*: A foreign key to the language dimension table to represent the “to” part of the third language pair for this translator
- *AggregateRateTimeliness*: An integer value to represent the average rating for the translator with respect to the timeliness of delivery
- *AggregateRateQuality*: An integer value to represent the average rating for the translator with respect to the quality of delivery
- *AggregateRatePricing*: An integer value to represent the average rating for the translator with respect to the pricing of delivery
- *AggregateRateCommunication*: An integer value to represent the average rating for the translator with respect to the communication during the process
- *Active*: a Boolean indicating whether this translator is still active or not.


DimTranslator	
	TranslatorKey
	GeographyKey
	NativeLanguageKey
	FirstLanguagePairFrom
	FirstLanguagePairTo
	SecondLanguagePairFrom
	SecondLanguagePairTo
	ThirdLanguagePairFrom
	ThirdLanguagePairTo
	AggregateRateTimeliness
	AggregateRateQuality
	AggregateRatePricing
	AggregateRateCommunication
	Active

Figure 10 Translator Dimension [source: own]

It is important to note here that while few translators have listed more than three language pairs, the decision has been to limit the data warehouse columns to represent only three pairs. It also helped to take that decision that we had related questions in the interviews with the operations personnel and the analysis of the current data sources

### 4.3.3 Facts

Facts tables are the main containers of the variable data which are the main target of analysis by default.

#### 4.3.3.1 FactSales

The sales fact table is the center of our data warehouse, at least at this stage. It records the most important business event. It is the pillar for analysis and answering most of the analytical questions that were required must come across this table. The following attributes enable the table to satisfy the expected requirements:

- *SalesKey*: The primary key for the fact table.
- *CustomerKey*: A foreign key to the customer dimension to refer to the buyer
- *TranslatorKey*: A foreign key to the translator dimension
- *OrderDateKey*: A foreign key to the date dimension to represent the date the order started
- *DeliveryDateKey*: A foreign key to the date dimension to represent the date the order was delivered
- *AmountReceived*: float value to represent the total amount of money received from the customer
- *AmountPaid*: float value to represent the total amount of money paid to the translator
- *LanguageFromKey*: A foreign key to the language table to represent the original language of the document

- *LanguageToKey*: A foreign key to the language table to represent the target language for the document translation
- *DocumentURL*: The address of the document(s) subject to translation
- *DeliveryURL*: The address of the delivered document(s) after translation

FactSales	
	SalesKey
	CustomerKey
	TranslatorKey
	OrderDateKey
	DeliveryDateKey
	AmountReceived
	AmountPaid
	LanguageFromKey
	LanguageToKey
	DocumentURL
	DeliveryURL

Figure 11 Sales Fact [source: own]

Combining all the parts together to have the dimension model in Figure 12, it becomes really clear how more understandable this model is compared to the original data sources. The dimension model takes a star shaped schema where all the dimensions relate to the fact table.

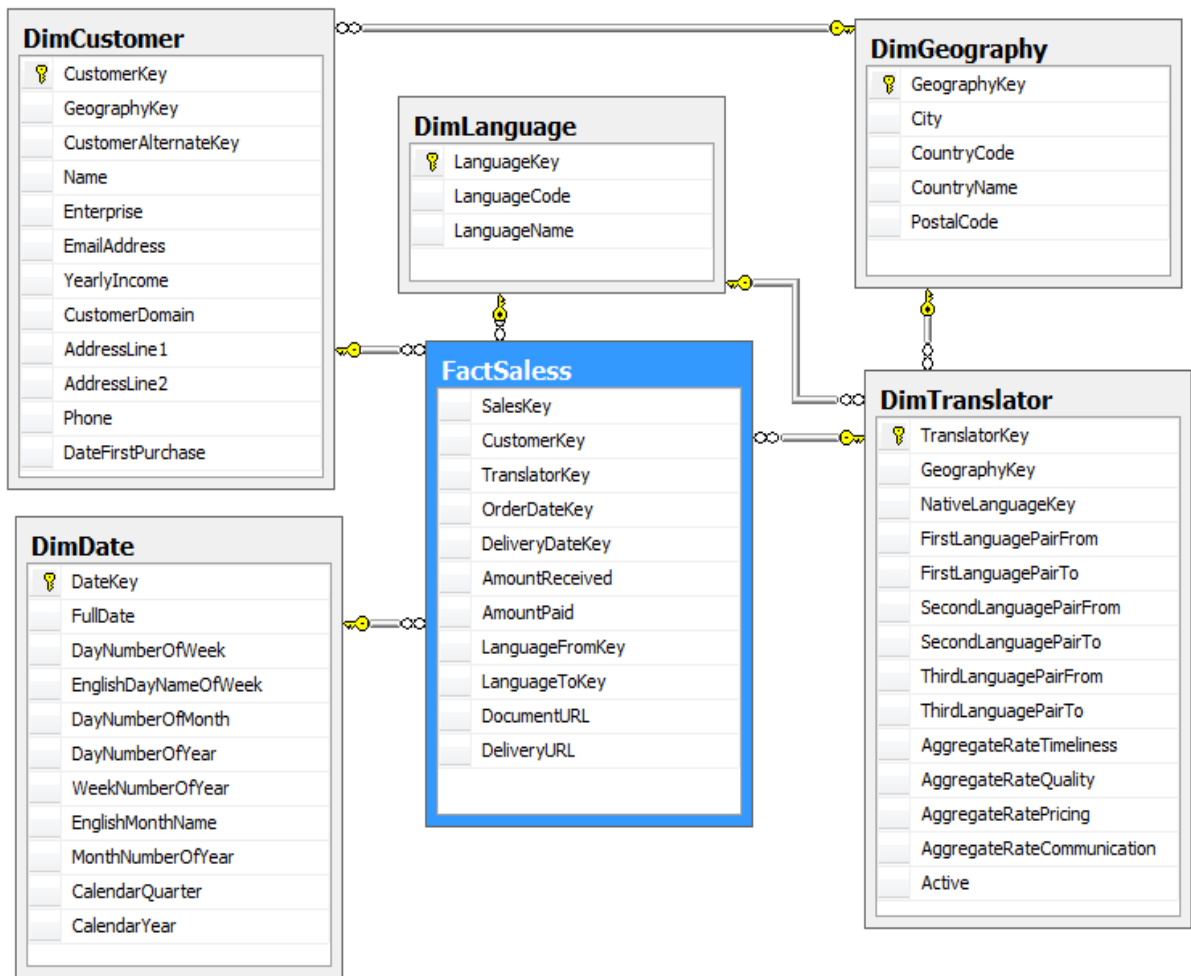


Figure 12 Dimension Model [source: own]

## 4.4 ETL (Extract, Transform, Load)

After having the dimensional model in place, the next step is to move the data to the data warehouse. Needless to say this is a very important phase in the project, for what good is any other step if the data didn't move to the data warehouse. Also, this step is very prone to error and errors at this stage would cost so much time and computational resources.

### 4.4.1 Etraction

The first step is **extracting** the data from the original data sources to temporary storage. Naturally, this would be scheduled to be executed in a time where the data sources aren't under heavy use, in order not to disrupt the operations. Also, keeping the data in a temporary storage means that in case some mistake happened in the process, there would be no need to repeat the extraction process and that we can carry on different experimentation without disrupting the original data sources again.

There are many ways to move the data sources to the temporary storage but the easiest on Microsoft SQL SERVER 2008 R2 is by directly copying the database from the current server to the new server, for each database individually, using SQL Server Management Studio SSMS (Microsoft, 2009).

1. First connect to the source server containing the databases

2. Right click on the first database and choose **Tasks-> Copy Database**,

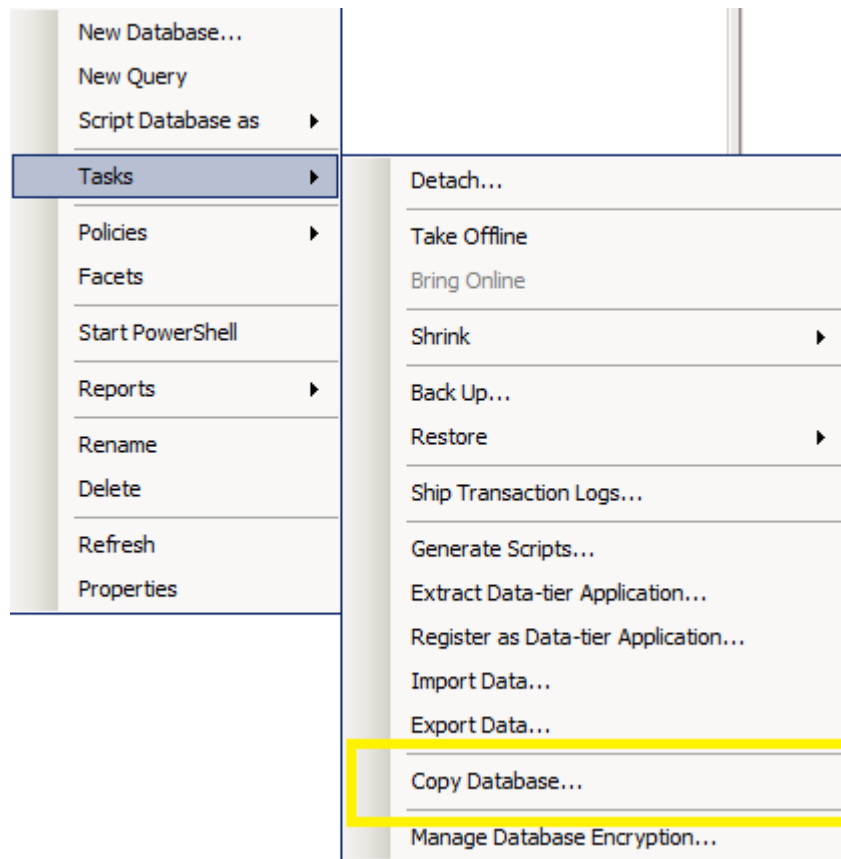


Figure 13: SSMS menu [source: own]

3. After clicking **Copy Database** a wizard window appears to choose the source server again and provide the login details as in Figure 14

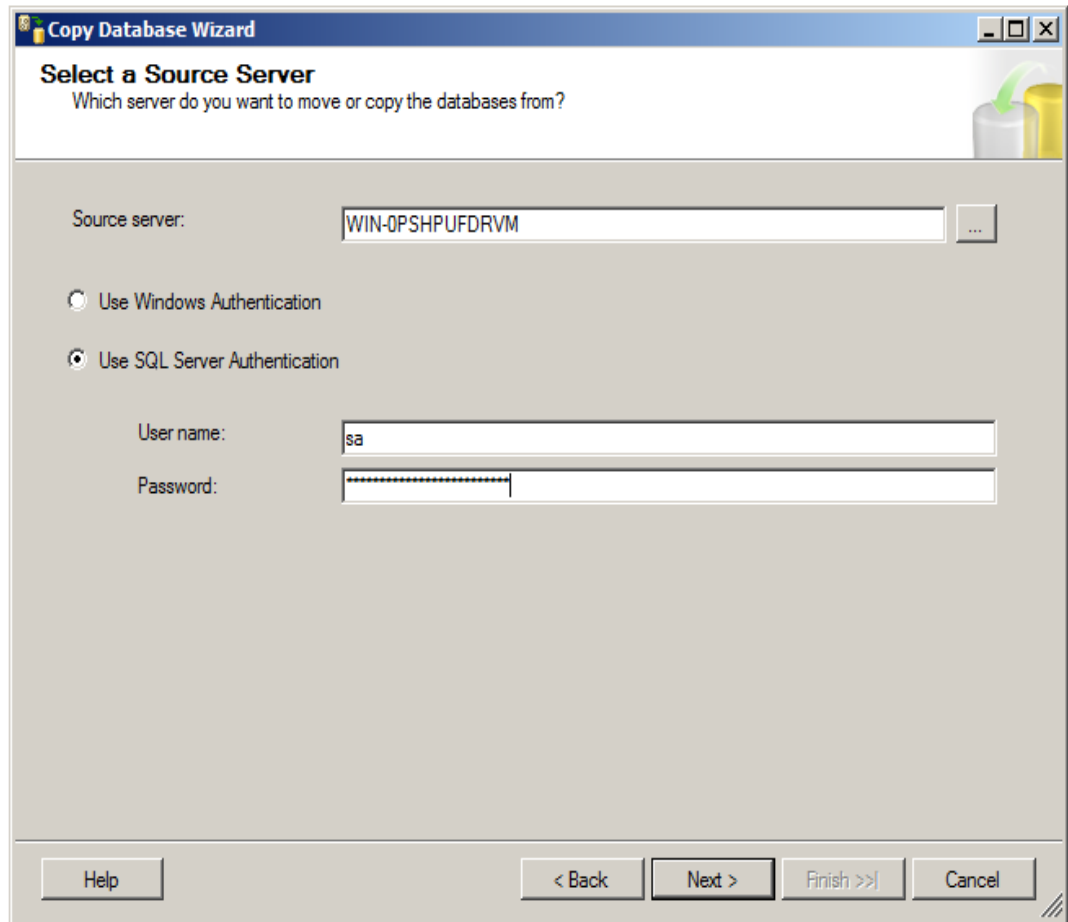


Figure 14: Copy Database First Step [source: own]

4. Then choose destination server **tmp\_ETL** and provide login details

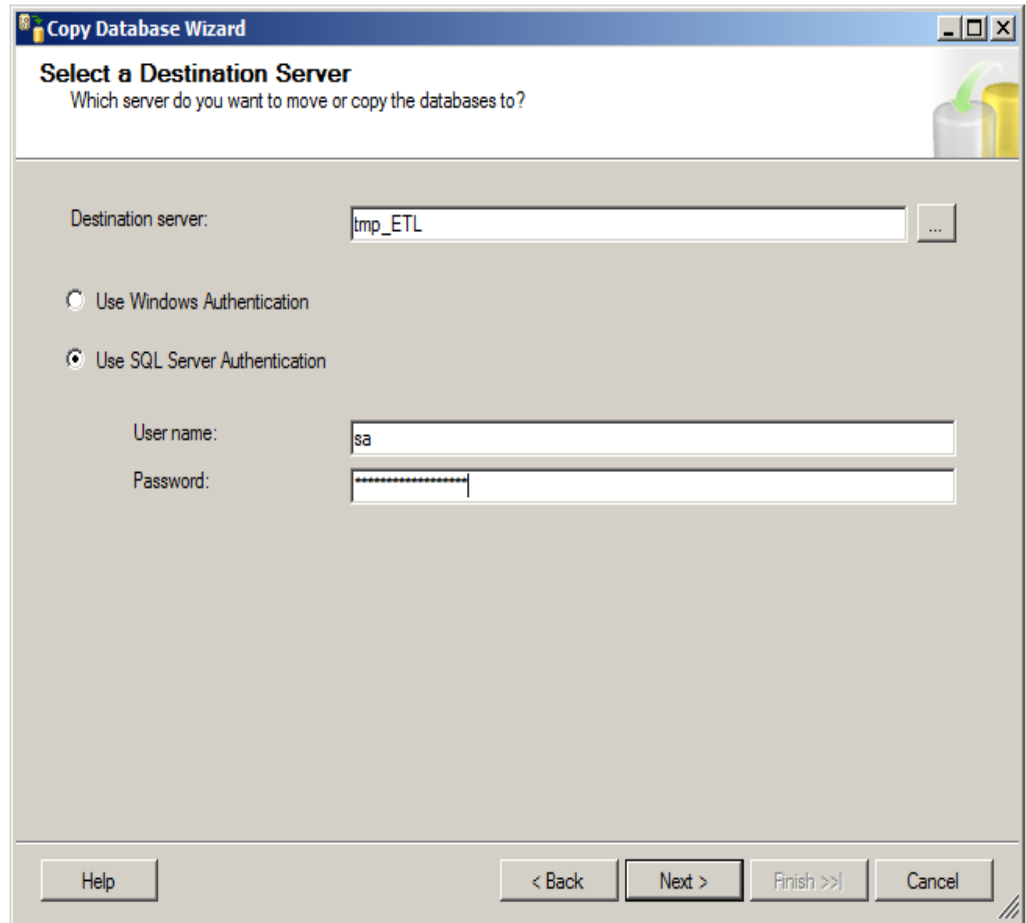


Figure 15 Copy destination [source: own]



5. The next step contains an important option to keep the data source online while the copying process is done in order not to disrupt the web applications

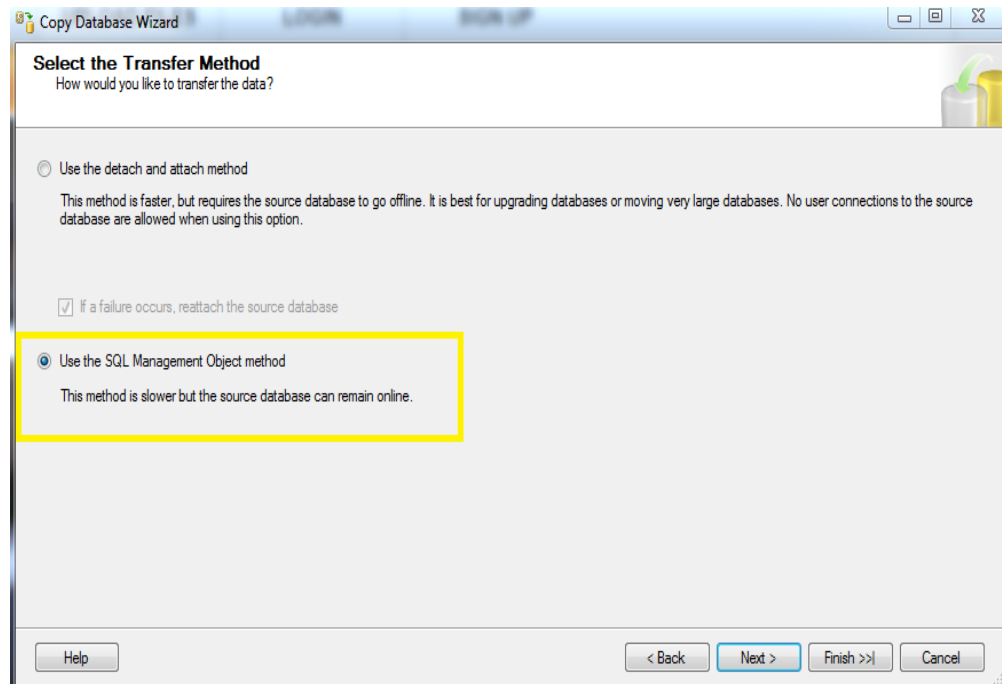


Figure 16 Database Online during copy [source: own]

6. After clicking finish the process will start copying the database.

The same steps would be repeated again for the **TranslatorAdvisor** database and then the temporary storage is ready for the transformation phase

#### 4.4.2 Transformation

Transformation phase is designing the process that would convert the current data sources to match the dimensional model, to flatten the nested relationships, to aggregate the facts and attributes, to ignore the unnecessary data...etc. The end product of this phase should be Comma-Separated-Values (CSV) files, each of them representing a table. Microsoft SQL Server Integration Service is the tool to be used for this task. The way it works is that the process is divided into parts each part is called a **Data Flow Task** and all the tasks are

contained in **Package** or **Control Flow**. The package could run as a whole or for each data flow task individually. Also, if one data flow task failed, the other tasks aren't affected and it could be decided what to do in case of failure; continue, stop the package or run another task flow. SSIS gives us the option to load the transformed data directly into the Microsoft based data warehouse but for the sake of comparing the loading performance, all data will be first converted to CSV files

Similar to how we designed the dimensional model we will again start with the dimensions and this case it is important to note that we will use the keys from the data source as the keys for the dimensions in order to facilitate the transformation process for the fact table and make it easier as well to generate reports later.

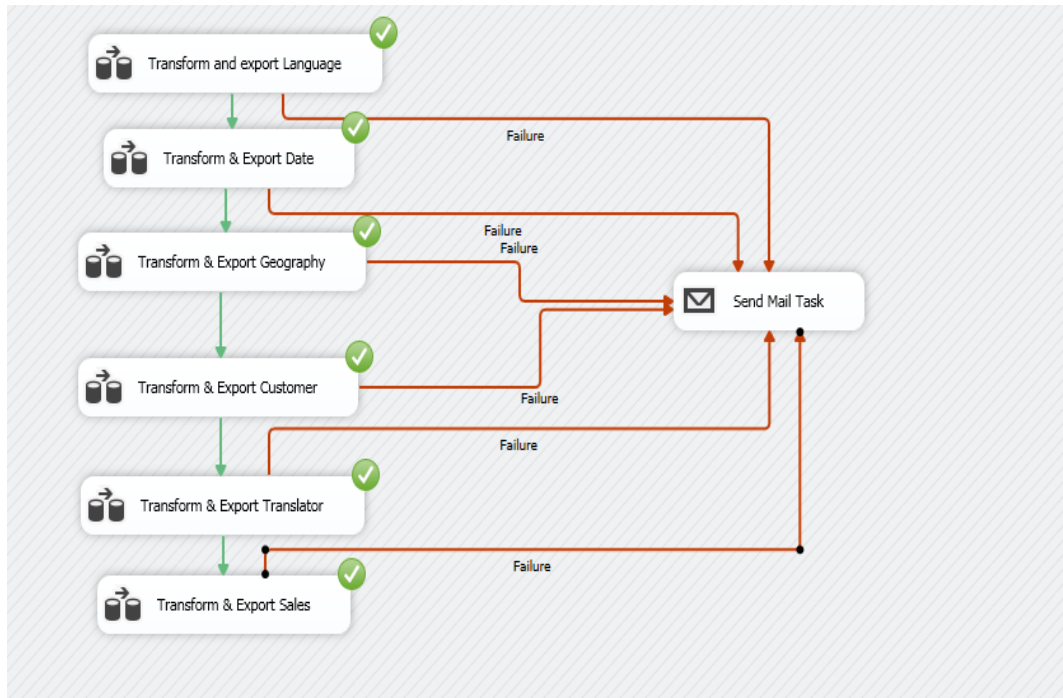


Figure 17 Transformation using SSIS [source: own]

In Figure 17 is the view of the package after the execution has ended. The green tick signs indicate that the data flow task executed successfully. The green lines connecting the data flow tasks indicates the action after the tasks was successful while the red lines indicate

the route after failure. In this case all the failure routes lead to an email task that would report the error and the failure log.

#### 4.4.3 Loading

Now that the data are exported into CSV files in the same structure as the dimensional model, the next step is loading the data into the warehouse and make sure it is ready for use. Loading the data into MS SQL Server is different from loading the data into Hadoop. Also, since The dimensional model was only implemented on MS SQL Server so far, there would be an extra step after loading the data on Hadoop to add the Hive meta-data, which the equivalent of the DDL, data definition layer, on MS SQL Server.

##### 4.4.3.1 Loading on Microsoft

There are many ways to upload the data into the data warehouse using SSIS (the same tool we used to export the data) or as usual, Microsoft provides an easy graphical interface for importing the data into the data warehouse in easy steps

- 1- Right click on the data warehouse in the SSMS and choosing **Import Data**

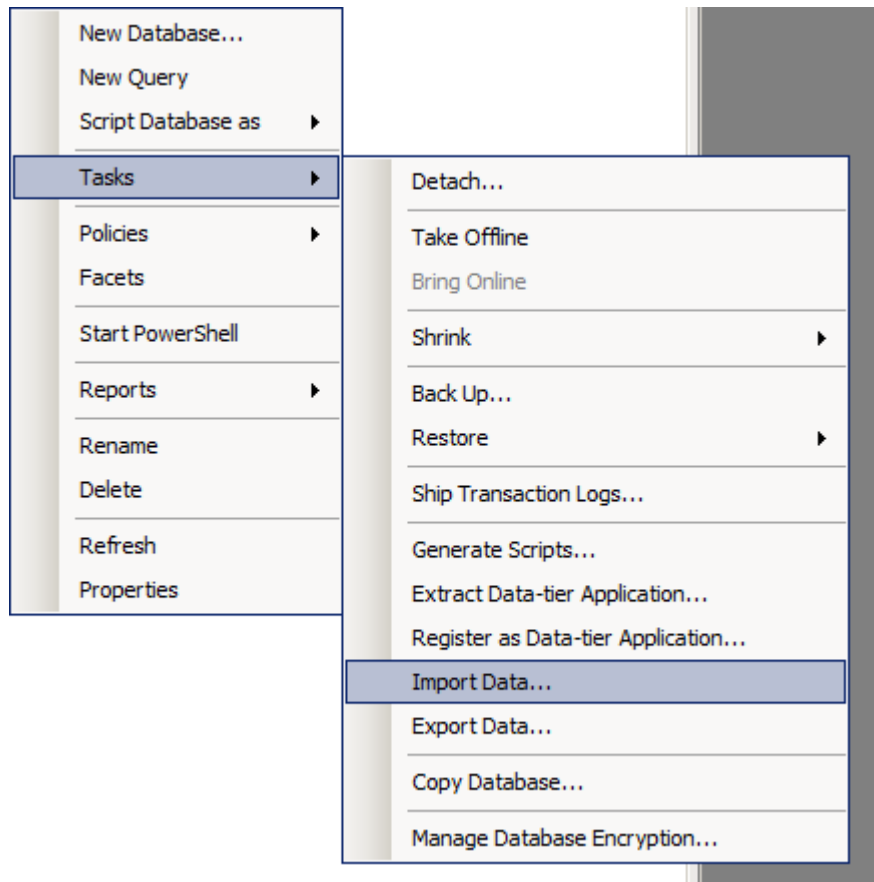


Figure 18 Import Data [source: own]

2- Since the files are in CSV format, the data source is a **Flat File Source**

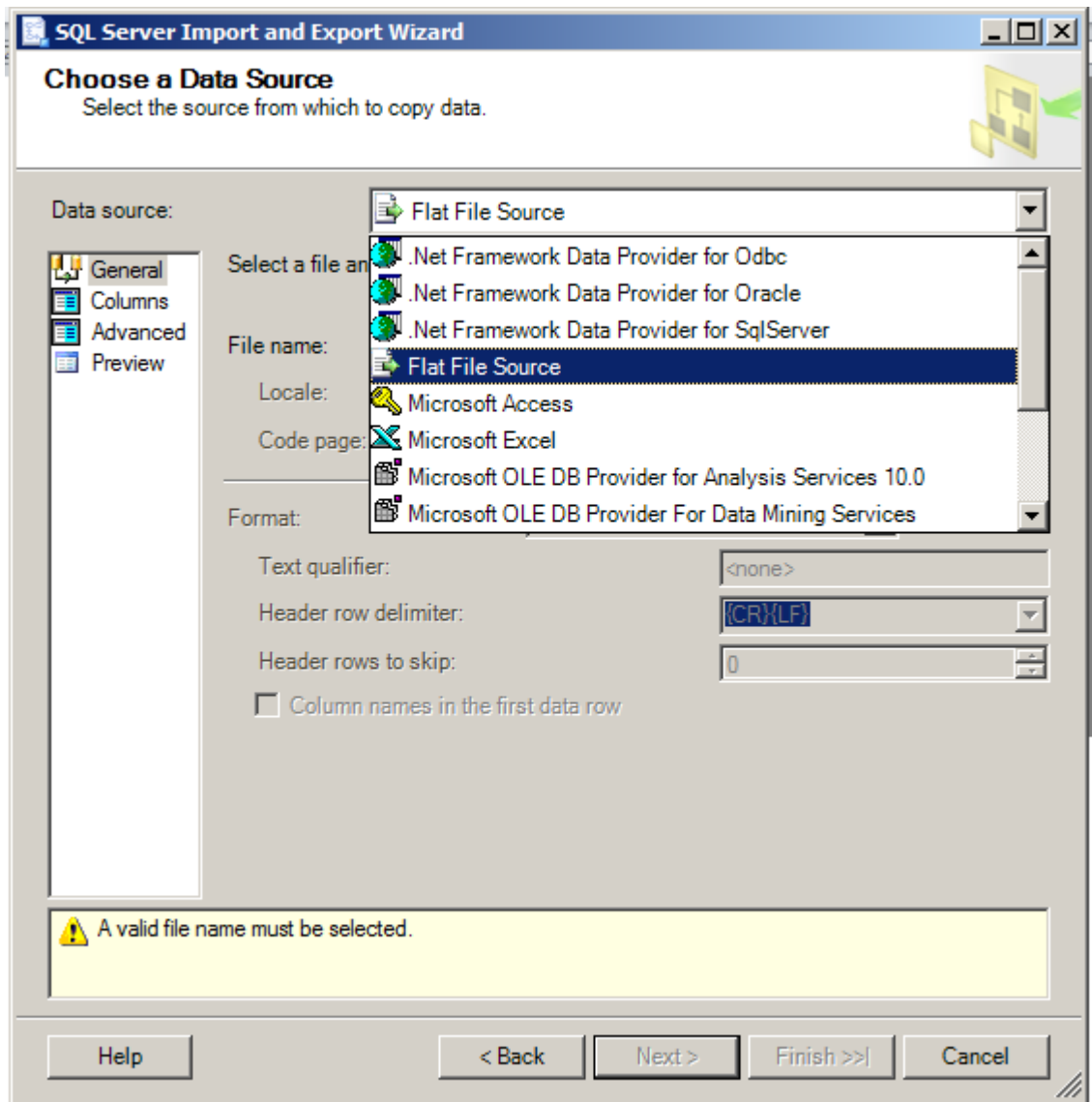


Figure 19 Loading data into SQL Server [source: own]

3- Choose the file then check the column mappings are correct

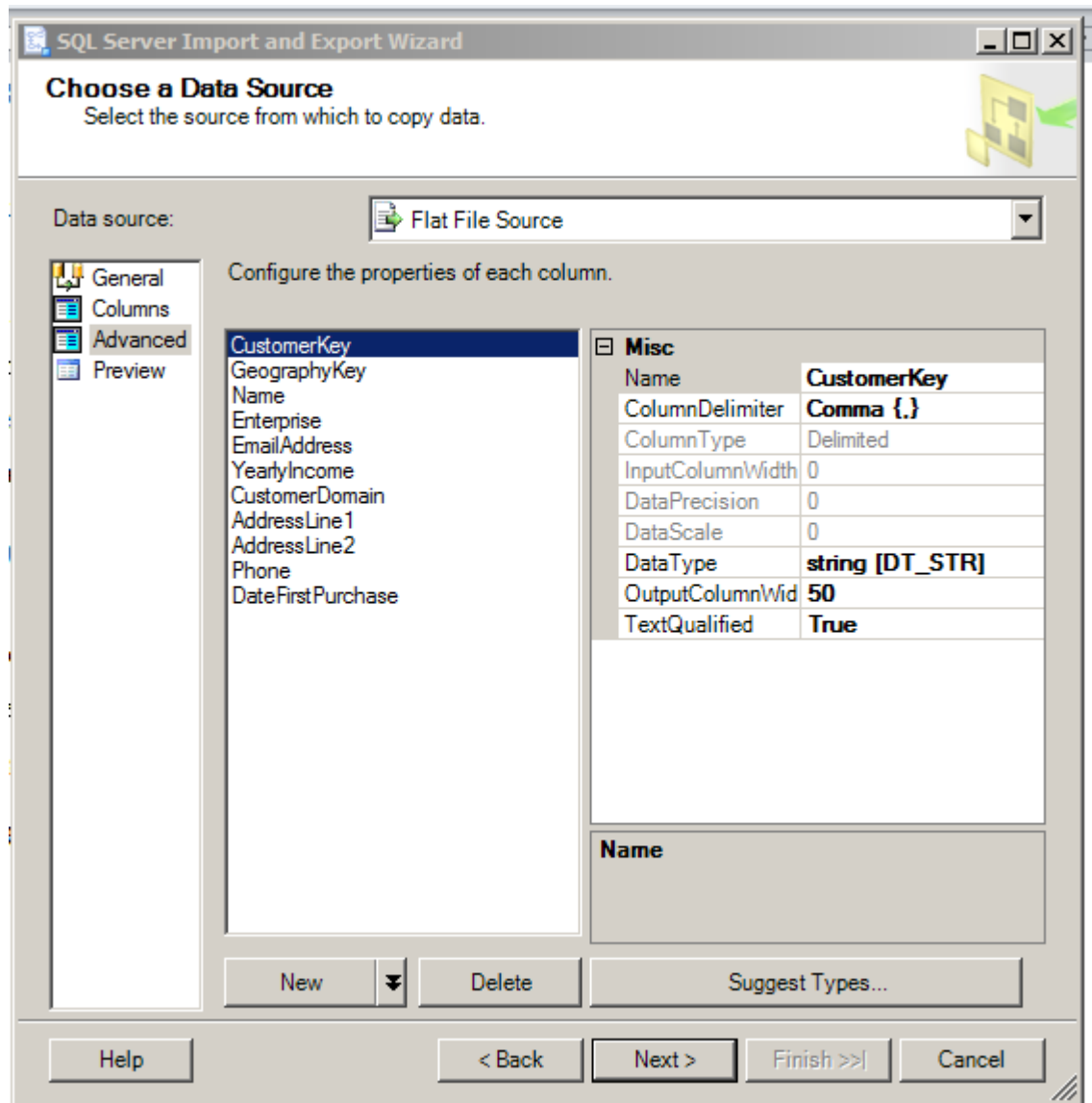


Figure 20 Input file and column mappings [source: own]

4- The continues with default options and loads the data in the corresponding table

The four steps above will be repeated for the dimension tables and fact table in the same order they are specified in the design to prevent foreign key errors

#### 4.4.3.2 Loading Data On Hadoop

Loading the data into Hadoop is much simpler than on Microsoft SQL Server since Hadoop will take the files directly. However, since, the data model isn't yet defined; Meta data would be defined for the tables after adding the files to HDFS, in order to make them independent from the files. Cloudera, a company developing open source solutions that are Hadoop based, developed a tool similar to SQL Server Management Studio (SSMS) that is also open source and free for use. Hue allows us to manage the whole cluster and also has a user management utility. Hue is browser based tool and through Hue graphical user interface uploading the data files is simply done as follows:

- 1- After logging in with the appropriate credentials

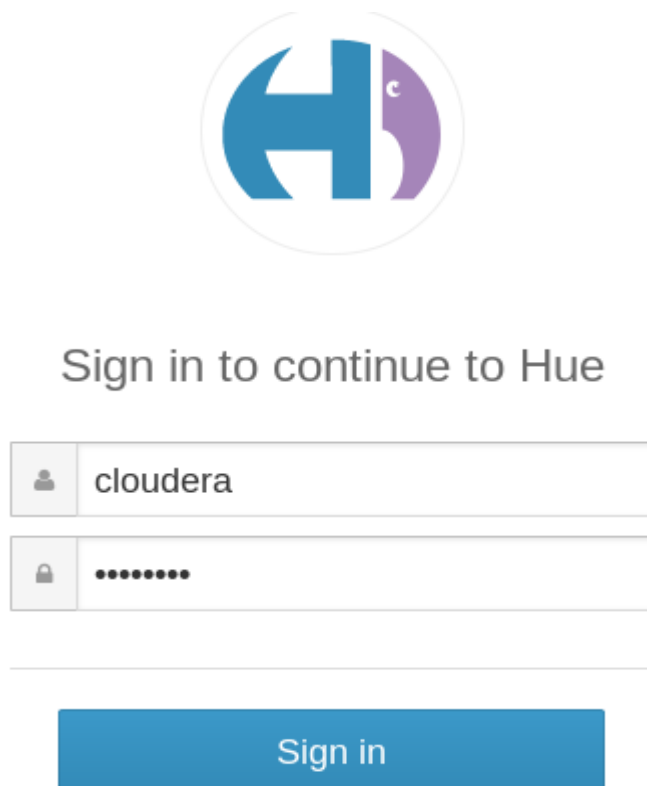


Figure 21 Hue Login Screen [source: own]

2- To upload the files and for HDFS management in general we click

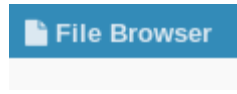


Figure 22 Hue file browser [source: own]

3- Just like on any file system, on HDFS the files would be placed in a newly created directory and give it a name “warehouse”

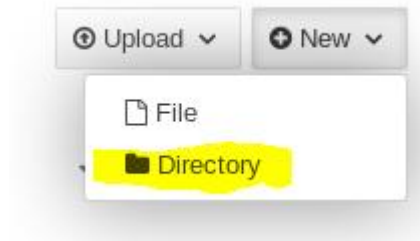


Figure 23 Hue New directory [source: own]

4- Hive tables can only be attached to directories not files so we will create a separate directory for each table. This also allows adding more data easily later on by just adding the new records in separate files without the need for merging

5- The next step is to upload the files

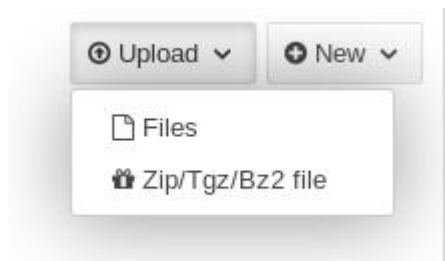


Figure 24 Hue Upload Files [source: own]

After the files are chosen



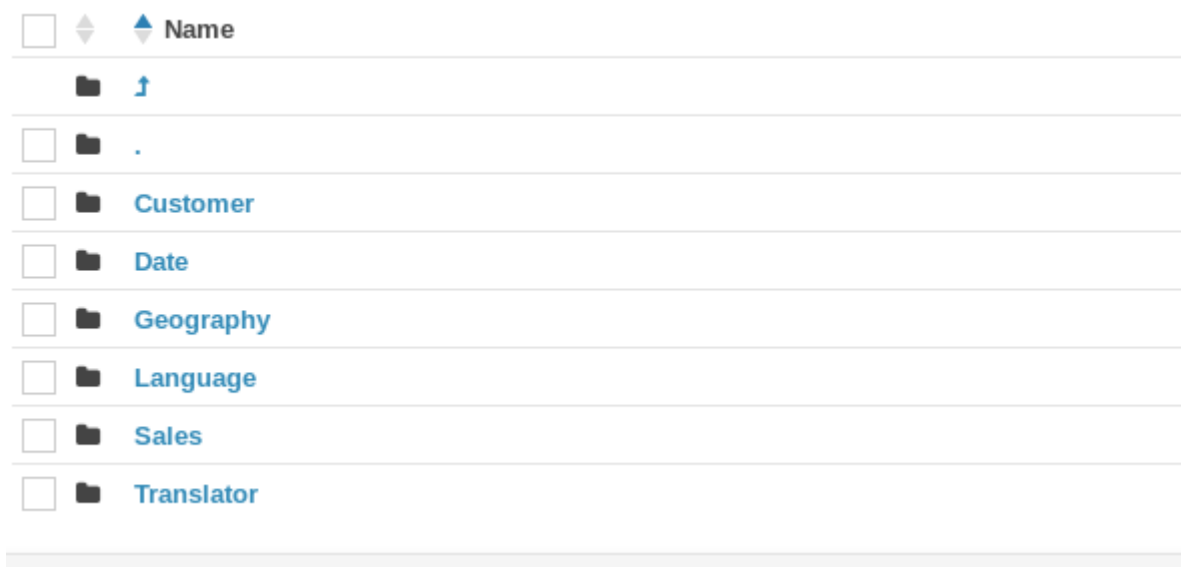


Figure 25 Hue: files added into HDFS [source: own]

Now that the files are in HDFS, the next step is to define the tables in Hive and make them point to their files respectively. The in Hive is very similar to the syntax of MySQL, both for DDL and for querying. The dimensions could be added in any order with no restriction regarding foreign keys as was the case with Microsoft SQL Server. In Figure 26 is the syntax for creating the customer dimension table that refers to DimCustomer.csv file.

```

1 create external table DimCustomer (CustomerKey int, GeographyKey int,
2                                     CustomerAlternateKey string, Name string,
3                                     Enterprise boolean, EmailAddress string,
4                                     CustomerDomain string, AddressLine1 string,
5                                     AddressLine2 string, Phone string, DateFirstPurchase DATE)
6 ROW FORMAT DELIMITED
7 FIELDS TERMINATED BY ','
8 location '/user/cloudera/datawarehouse/Customer'
9 tblproperties ("skip.header.line.count"="1");

```

Figure 26 Hue: Creating a table [source: own]

There are few things to note about the syntax for creating the table:

- 1- The keyword **external** means that Hive will not copy this data file into its own folders and will just maintain an separate layer for meta-data

- 2- The **ROW FORMAT** part and the **FIELD TERMINATED BY** parts are optional and are usually used with text data format, as is the case now. Also, they provide the flexibility to read different text formats like TSV, TAB, logs or other formats. Also, Hive provides a way to define shortcuts for format specifications to eliminate repetition and centralize changes later on.
- 3- The last line defines the number of rows to skip to 1 because in our files the first line contains the column headers which is already defined in the table creation query along with their data types. This is necessary because the table names would also violate the data types of the columns.

To confirm that the step was executed correctly first we should check that the file still exist in the same directory and then make sure that Hive can query the data from that file.

To query the table for a sample of 100 rows, we use a SQL-like syntax in Hive:

```
SELECT * FROM dimcustomer LIMIT 100
```

The result of this query is shown in Figure 27

	dimcustomer.customerkey	dimcustomer.geographykey	dimcustomer.customeralternatekey	dimcustomer.name	dimcustomer.enterprise	dimcustomer.emailaddress	dimcustomer.age
0	11000	26	AW00011000	Jon	False	jon24@adventure-works.com	9000
1	11001	37	AW00011001	Eugene	False	eugene10@adventure-works.com	6000
2	11002	31	AW00011002	Ruben	False	ruben35@adventure-works.com	6000
3	11003	11	AW00011003	Christy	False	christy12@adventure-works.com	7000
4	11004	19	AW00011004	Elizabeth	False	elizabeth5@adventure-works.com	8000
5	11005	22	AW00011005	Julio	False	julio1@adventure-works.com	7000
6	11006	8	AW00011006	Janet	False	janet9@adventure-works.com	7000
7	11007	40	AW00011007	Marco	False	marco14@adventure-works.com	6000
8	11008	32	AW00011008	Rob	False	rob4@adventure-works.com	6000
9	11009	25	AW00011009	Shannon	False	shannon38@adventure-works.com	7000
10	11010	22	AW00011010	Jacquelyn	False	jacquelyn20@adventure-works.com	7000
11	11011	22	AW00011011	Curtis	False	curtis9@adventure-works.com	6000

Figure 27 Hue: Sample customer data [source: own]

Similarly, meta-data will be defined for the other tables until we have the whole dimensional model defined similar to how it is on Microsoft. After this is done, we will try a query that answers a one of the analytical questions. For example, this is a query to answer the question: Which customers are from Australia

```
SELECT * from DimCustomer c, DimGeography g
WHERE c.geographykey = g.geographykey
and g.countryname = 'Australia'
```

The query above joins two tables and uses a filter on one of them to return a specific set that matches the conditions specified by the question. Hive also provides details on the implementation plan that lead to the result. This plan could be a powerful tool for debugging more complex queries.

Another way to process and analyze data on Hadoop is Pig which has syntax similar to Scala but it doesn't benefit from the meta-data for the files so the script deals with the file on its own. This method is more suitable when the data is not structured and there is no way to define a table-like structure for the data.

## 4.5 Implementation Remarks

The implementation of the data warehouse in both technologies has been an exciting and challenging journey. Starting from the dimensional design, it became clear how in general the data warehouse can benefit the firm and enhance the reporting capabilities. Moving to implementation on both technologies the differences became clear in some of the comparison points. Based on the criterion that was specified in the methodology the following table offers the comparison between the two technologies.

Table 5 Final Assessment [source: own]

Criteria	Microsoft SQL Server	Apache Hadoop
Capacity	Ideal for Small, medium and moderately large amounts of data. Not suitable for big data. Also can only handle structured data	Ideal for Big Data usually more than 10 billion data points. Great for unstructured and unusual data formats. However, for small data amounts, the over head outweighs the concurrency benefits
Loading Performance	Loading data on SQL server is a demanding process and requires loading the table in specific order to maintain foreign key constraints	Loading data on Apache Hadoop is as simple as copying data files from one disk to another, doesn't require any order for copying the data and is only limited by the IO speed of the underlying hardware

<p><b>Client/Server connectivity:</b></p>	<p>It is possible to access the data warehouse from almost every major tool. However, each connection takes up resources which could lead to connection termination if the number of connections is too large, even if the connection is not actually making any data calls</p>	<p>Connection can be established through a REST API which decreases the amount of resources needed for each open connection and makes connecting to the data warehouse available even from a command prompt</p>
<p><b>Query Performance</b></p>	<p>For the capacity range suitable for Microsoft SQL Server, it offers powerful performance on queries and makes use of indexing, caching and other tools. However, for Big Data the performance drops.</p>	<p>Offers powerful querying performance for huge amounts of data by using concurrency and data locality. However, for small amounts of data the over head for starting MapReduce jobs outweighs the performance gains by a lot.</p>

## 5. Conclusion

Data warehouses offer businesses great analytical abilities when the business is generating more and more data but it also comes with a considerable investment of money, time and human resources. So, the decision to start a data warehouse or not should first be evaluated from a business point of view. This project had a main goal to provide a comparison between Microsoft SQL Server and Apache Hadoop for creating data warehouses. The comparison didn't aim to eliminate one of the two options in favor of the other but rather uncover the decision process when it is necessary to make that decision. To satisfy this purpose, the whole process of creating a data warehouse was executed for the two technologies, following state of the art design techniques and explaining the choices made for the design and implementation. The comparison is based on:

- Capacity: the amount of data that could be loaded without affecting functionality,
- Loading the data: the ease and performance of the process
- Connectivity: the ease and versatility of connecting to the data warehouse
- Querying: the ease and performance of querying the data in the data warehouse

### 5.1 Apache Hadoop

Apache Hadoop as with most of the major open source projects offers an opportunity to break free of the licensing fees, the exclusiveness within the environment to a certain company's chain of products. However, as with most of the open source projects as well, is still in need of development and support to become a full replacement. Apache Hadoop is an exception of this case because it received a lot of support from the very beginning both technical and financial from many of the sector leaders; with companies like Google, Oracle, IBM and Cloudera investing more and more into building a full scale environment and enhancing the overall use of the technology.

Hadoop is extremely flexible with the data formats being added because it all goes down as storing a file on a file system, HDFS in this case. Also, loading the data into HDFS is

really fast and is only limited by the read/write speed of the hard disks. Hadoop is also cheaper and easily extensible and can handle virtually any amount of data. Querying is slower on Hadoop when dealing with small amounts of data so the decision to use Hadoop isn't always the correct decision. Hadoop is cheap and many of companies started offering Hadoop on the cloud and thus eliminating the need of a bulk investment in hardware in the beginning and the hassle of server management.

## 5.2 Microsoft SQL Server

Microsoft SQL Server is a reliable and widely used database management system. Even though it is not cheap in terms of acquisition and licensing, it is suitable for medium and big enterprise with manageable amounts of data. Also, Microsoft provides a complete development ecosystem and a variety of interconnected solutions like SQL Server Integration Services SSIS, SQL Server Reporting Services SSRS and SQL Server Analytics Service SSAS and all of these services are integrable with each other and with the other Microsoft technologies like ASP.NET and fits really well with all of it with special components developed by Microsoft and other third parties to make everything work together nicely

Microsoft SQL Server is suitable for structured data up to very large amounts but the limit is much less than Hadoop. Loading the data into SQL Server is easy thanks to the graphical wizards provided by Microsoft but again would normally be slower than HDFS because of the foreign key checks and other insertion overhead. SQL Server is easily connectable with almost all the major technologies and tools either using built in features or using adapters. Querying on SQL server is done using SQL queries or stored procedures and is highly efficient thanks to features like indices and caching.

The decision to use Apache Hadoop or Microsoft SQL Server depends on many factors that relate to the business and the nature of data. Answering questions like: How much data is there? Is it structured, semi-structured or unstructured data? What is the current technology stack in use for the rest of the organization? What are the resources allocated for the data warehouse? Answering each of these questions would be a step towards the right decision

## References

1. DEVLIN, B. A. and MURPHY, P. T. 1988. An architecture for a business and information system. New York : IBM, 1988, IBM Systems Journal, Vol. 27, pp. 60-80. ISSN: 0018-8670.
2. Apache Software Foundation. 2016. Apache Hadoop YARN. Apache Hadoop. [Online] January 16, 2016. [Cited: February 02, 2016.] . Available at: <http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>.
3. BULUSU, Lakshman . 2012. Open Source Data Warehousing and Business Intelligence. Boca Raton : CRC Press, 2012. ISBN 9781439816400.
4. BUTLER, Andrew and STRANGE, Kevin. 2002. A Data Warehouse Evaluation Model. Stamford, USA : Gartner, 2002. pp. 1 - 4. G00109604.
5. CORR, Lawrence and STAGNITTO, Jim. 2011. Agile Data Warehouse Design: Collaborative Dimensional Modeling, from Whiteboard to Star Schema. 1st. Leeds : DecisionOne Press, 2011. ISBN: 0956817203.
6. DEVLIN, Barry. 1996. Data Warehouse: From Architecture to Implementation. Dublin : Addison-Wesley Professional, 1996. ISBN: 0201964257.
7. DIJCKS, Jean-Pierre. 2014. Updated: Price Comparison for Big Data Appliance and Hadoop. The Warehouse insider. [Online] April 03, 2014. . Available at: [https://blogs.oracle.com/datawarehousing/entry/updated\\_price\\_comparison\\_for\\_big](https://blogs.oracle.com/datawarehousing/entry/updated_price_comparison_for_big).
8. Genomics England. 2014. The 100,000 Genomes Project. Genomics England. [Online] UK Department of Health, July 21, 2014. [Cited: October 10, 2015.] . Available at: <http://www.genomicsengland.co.uk/the-100000-genomes-project/>.
9. GROVER, Mark, et al. 2015. Hadoop Application Architectures. Sebastopol : O'Reilly Media, 2015. ISBN: 1491900083.
10. HOLMES, Alex. 2014. Hadoop in Practice. 2nd. New York : Manning Publications, 2014. ISBN: 1617292222.
11. INMON, William H. 1992. Building the Data Warehouse. New York : John Wiley & Sons, Inc., 1992.
12. MISHRA, Priti and Eich, Margaret H. 1992. Join Processing in Relational Databases. Dallas : ACM Computing Surveys, 1992, Vol. 24.



13. KIMBALL, Ralph and Ross, Margy. 2013. The Data Warehouse Lifecycle Toolkit. Indianapolis : John Wiley & Sons, Inc., 2013. ISBN: 1118530802.
14. KIMBALL, Ralph, Mundy, Joy and Thornthwaite, Warren. 2011. The Microsoft Data Warehouse Toolkit. Indianapolis : Wiley Publishing, Inc., 2011. ISBN:978-1-118-06795-6.
15. KRISHNAN, Krish. 2013. Data Warehousing in the Age of Big Data. Waltham : Morgan Kaufmann, 2013. ISBN: 9780124058910.
16. LAM, Chuck. 2011. Hadoop in action. 1st. Greenwich : Manning Publications, 2011. ISBN:1935182196.
17. LEVERENZ, Lefty. 2016. HCatalog: Using HCatalog. Apache Software Foundation. [Online] January 24, 2016. . Available at: <https://cwiki.apache.org/confluence/display/Hive/HCatalog+UsingHCat>.
18. MICROSOFT. 2009. Create a Full Database Backup (SQL Server). Microsoft technical network. [Online] Microsoft, January 12, 2009. [Cited: November 12, 2015.] . Available at: [https://msdn.microsoft.com/en-us/library/ms187510\(v=sql.105\).aspx](https://msdn.microsoft.com/en-us/library/ms187510(v=sql.105).aspx).
19. ORACLE. 2016. Oracle Technology Global Price List. Oracle: Software Investment Guide. [Online] January 21, 2016. . Available at: <http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf>.
20. SPICER, Drag, et al. 1998. Timeline of Computer History. Computer History Meuzeum. [Online] Computer History Museum, December 2, 1998. [Cited: November 16, 2015.] . Available at: <http://www.computerhistory.org/timeline/memory-storage/>.
21. WHITE, Tom. 2015. Hadoop: The Definitive Guide. s.l. : O'Reilly Media, 2015.
22. YOUTUBE. 2015. Statistics. Youtube. [Online] 2015. [Cited: November 1, 2015.]. Available at: <https://www.youtube.com/yt/press/statistics.html>.