Jonny Nyman

Data Integration

Steps towards an efficient and user-friendly process

Master’s thesis in Information System
Supervisor: Jozsef Mezei
Faculty of Social Sciences, Business and Economics
Åbo Akademi

Åbo 2019
Abstract:

As new business technology has developed exponentially in the last two decades, data-driven decision making in organizations has become a key factor to gain a competitive advantage on the market. Instead of relying on intuitive decision making or decisions based on one’s own working experience and observations, decisions based on data have become a fundamental foundation for making the right business decision, at the right time. To provide organizations with accurate and usable data for decision making and analytics, a complicated and resource consuming data integration process needs to be performed, to collect all the data available in disparate sources, and to ensure the quality of the data that is used.

To increase the awareness of the importance of the data integration process and its complexity, and to answer the research questions of this master’s thesis, a literature review in the subject area, in addition to a real-world data integration process has been performed at the case company in hand. The thesis will also cover necessary theory of data warehousing, databases and business intelligence, that are vital regarding data integration. The thesis will highlight difficulties, different approaches and best practices suggested by authors in the literature.

The results in the thesis present a solution that the case company can use to make its data integration process more user-friendly and efficient. In addition, an informal interview with employees working with data integration at the case company is done to back-up the result. The thesis also highlights what organizations need to consider when designing an organizational data integration process.

Key words: Data Integration, Data Warehouse, Business Intelligence, ETL
Table of Contents

1 Introduction .................................................................................................................. 1
  1.1 Case Company ........................................................................................................ 2
  1.2 Problem .................................................................................................................. 3
  1.3 Motivation ............................................................................................................... 5
  1.4 Research Questions ............................................................................................... 5
  1.5 Outline ..................................................................................................................... 6

2 Theory and Literature Review ....................................................................................... 7
  2.1 Database Management and Design ......................................................................... 7
    2.1.1 Databases ......................................................................................................... 7
    2.1.2 SQL ............................................................................................................... 9
    2.1.3 Stored Procedure ............................................................................................ 10
    2.1.4 Metadata ....................................................................................................... 11
    2.1.5 Data Warehousing ......................................................................................... 12
  2.2 Data Integration ....................................................................................................... 22
    2.2.1 Data Integration Requirements ...................................................................... 24
    2.2.2 Business Requirements ............................................................................... 25
    2.2.3 Data Profiling .............................................................................................. 26
    2.2.4 Extract Transform and Load ......................................................................... 27
    2.2.5 Manual Coding versus Tool-Based Data Integration .................................... 34
    2.2.6 Testing .......................................................................................................... 36
    2.2.7 Scheduling .................................................................................................. 37
  2.3 Summary of the Literature Review .......................................................................... 37

3 Research Methodology ................................................................................................ 40

4 Case Description ........................................................................................................ 44
  4.1 Business Requirements ......................................................................................... 44
  4.2 Data Profiling ....................................................................................................... 45
  4.3 Extract .................................................................................................................. 47
  4.4 DataStage ............................................................................................................ 48
    4.4.1 Pipeline Execution Id ...................................................................................... 50
    4.4.2 Extraction Population Id ............................................................................... 52
  4.5 Transformation .................................................................................................... 54
  4.6 Loading .................................................................................................................. 59
4.7 Testing........................................................................................................................................ 60
4.8 Scheduling.................................................................................................................................. 61
5 Findings and Discussion .................................................................................................................... 63
5.1 Data Integration Process at IF Insurance in Relation to the Literature. ........................ 63
5.2 Findings in the Literature .............................................................................................................. 63
5.2.1 What are the Most Relevant Issues to Be Considered when Designing an Organizational Data Integration Process? ......................................................... 65
5.3 Case Company Analysis .................................................................................................................. 65
5.3.1 Extract Population Id and Pipeline Execution Id ................................................................. 65
5.3.2 The New Solution from a Developer’s Point of View ............................................................. 67
5.3.3 Performance ............................................................................................................................. 68
5.3.4 Drawbacks with the Solution .................................................................................................... 69
5.3.5 Code Repository ...................................................................................................................... 69
5.3.6 Drawbacks and Challenges with Creating a Code Repository ............................................... 70
5.3.7 How Can Organizations Improve Traditional ETL-processing in Terms of Efficiency and User-friendliness? ................................................................. 70
6 Final Remarks .................................................................................................................................. 72
6.1 Limitations of the Thesis ............................................................................................................... 72
6.2 Further Research ............................................................................................................................ 72
7 Svensk Sammanfattning .................................................................................................................... 74
7.1 Teori .............................................................................................................................................. 74
7.1.1 Datalager .................................................................................................................................. 75
7.1.2 Extrahera, Förvandla, Ladda .................................................................................................. 75
7.1.3 Testning och Schemaläggning ................................................................................................. 76
7.2 Metod ............................................................................................................................................ 76
7.3 Dataintegrationsprocessen hos IF Skadeförsäkring ..................................................................... 77
7.4 Resultat och diskussion ............................................................................................................... 78
8 References: ...................................................................................................................................... 79

Figure 1, SQL example......................................................................................................................... 10
Figure 2, Stored procedure .................................................................................................................. 11
Figure 3, federated data warehouse architecture based on illustration by Ponniah (2011). 15
Figure 4, Bus data warehouse architecture based on illustration by Ponniah (2011) .................. 16
Figure 5, ETL example created in DataStage ................................................................................. 28
Figure 6, simplified ELT example ..................................................................................................... 29
Figure 7, Ndp.Animal table definition ............................................................................................ 45
Figure 8, Ndp.Animal table definition................................................................. 47
Figure 9, Sequence job in DataStage................................................................. 49
Figure 10, JEL_WPExp_Waypoint_Exposure....................................................... 49
Figure 11, metadata sequence job.................................................................... 50
Figure 12, Pipeline execution metadata generating step part 1.......................... 51
Figure 13, Pipeline execution metadata generating step part 2............................ 51
Figure 14, Extc_Pop_Id metadata generating step part 1.................................... 52
Figure 15, Extc_Pop_Id metadata generating step part 2.................................... 53
Figure 16, source data....................................................................................... 54
Figure 17, execute a stored procedure............................................................... 57
Figure 18, removing of duplicates.................................................................... 57
Figure 19, creating one row............................................................................... 58
Figure 20, Coalesce function........................................................................... 58
Figure 21, Result sample................................................................................... 60
Figure 22, Run status of the pipeline................................................................. 62

Table 1, literature review summary.................................................................. 37
Table 2 Example of transformations within DataStage........................................ 51
Table 3 transform - rename............................................................................. 53
Table 4 Example of metadata generated in a stored procedure......................... 55
1 Introduction

The success of many organizations depends on their decision making. The right price and quality of products are, without doubt, factors that will lead a business towards success, but a decision to market a product to the wrong customer group can have the opposite effect. Business intelligence that, according to Dayal, Castellanos, Simitsis and Wilkinson (2009), is a set of tools and practices for gathering, integrating, analyzing, and presenting large volumes of information to enable better decision making. Van (2012) states that business intelligence strives to support businesses in decision making with providing the right data at the right time and form to support decision making.

There is a wide range of different kinds of reporting and analytical tools available on the market. Reporting tools in most cases focus on descriptive analytics, as Sivarajah, Kamal, Irani, and Weerakkody (2017) mention, to describe and present data for its user by using simple statistical methods such as standard deviation, median, averages and aggregated values. Van (2012) writes that analytical tools tend to use algorithms and data mining to support predictive analytics. Gandomi and Haider (2015) state predictive analytics is used to study patterns in data to predict the outcome in the future.

The value of these tools largely depends on the availability and quality of the data acquired by the organization. Technically, it is possible to connect a reporting or analytical tool to a production database that is used in the daily processing of transactions (Van 2012). However, data stored in production databases are, in most cases, not in the format that is optimal to support the end user. Normally, a user would like to have all the customer data in one table or view. In practice, customer data can be stored in several different sources and tables containing different attributes. Creating a report on the data stored in a production database containing missing and defective data is generally not a good option. While many tools have the ability to modify data on the go, running queries from multiple sources while
performing transformations on the data will result in slow performance when queries tend to be complex. Another issue is that it can also be difficult to guarantee the consistency of the reports created when different sources and tools can have different transformation rules applied. In addition, querying the production database can be so input and output (I/O) intensive that users entering data into the database can encounter performance issues. In some cases, Van (2012) notes, even historical data can be missing in the source table due to storage issues and therefore descriptive analytics cannot be performed. To overcome these issues business intelligence tools tend to be used upon a data warehouse where the data is stored in the right format and of the right quality needed for creating reports and performing analytics.

To ensure that the data in the data warehouse are of good quality and to avoid issues with consistency and defective data, there is a vital data integration process that provides the business with the data it needs. This master’s thesis will analyze literature in this field and conduct a data integration process from start to finish to evaluate how this process can be improved overall and for the company in question.

1.1 Case Company

IF insurance is a property and casualty (P&C) insurance company fully owned by Sampo plc, listed on NASDAQ OMX Helsinki. IF insurance is the leading P&C insurance company in the Nordic countries, mainly operating in Sweden, Finland, Norway and Denmark. The company has approximately 3.7 million customers in the Nordic and Baltic countries, handling over 1.4 million claims annually. The company was founded in 1999 in a merger involving several companies. As a result, the company has data available in many different source systems which makes IF Insurance an excellent choice as case company for this master’s thesis. (IF Insurance 2019)
1.2 Problem

In the following, the most important problematic issues present in the company’s data handling and integration processes will be discussed.

**Software tool:** Currently most of the Extract, Transform, Load (ETL)-process is done inside IBM InfoSphere DataStage. The tool was acquired by IBM in 2005 and since then the software has not seen any major updates. DataStage is not a system one often sees as a requirement when applying for a job and the tool is not something one learns to use in the universities. This, of course, creates a problem when recruiting new employees to the data integration team. The learning curve for DataStage is steep and will require plenty of time before a new employee can start doing new development for the business. The new employee will also need assistance from a developer teaching how to use the tool to obtain the desired result. The learning period will require plenty of time from the developer and will result in the team delivering less new development during this time.

**Data format:** The data formats extracted from different sources tend to vary considerably. As an example, one can mention social security numbers that tend to be in different formats in different data sources. For instance, a Finnish security number can in some systems be inserted with a hyphen, for instance 101294-****, and sometimes without. This can be a typical problem when trying to map a social security number from two different sources. Another example is timestamps and ‘date’ data type, that can come in different formats. For example, the different formats YYYY/MM/DD or DD/MM/YYYY need to be formatted into timestamp with milliseconds.

**Error solving:** Errors and failed batch jobs in DataStage occur on a weekly basis. Sometimes the errors might be small and can be corrected within an hour but sometimes it can require even days to solve a problem. Most often the error will be found in an old DataStage job created several years ago by a developer who is no
longer working in the company. The error solving in a DataStage job filled with different transformations and logic without any comments explaining the process is a tough nut to crack. Sometimes it can even save the developer some time to recreate the whole job with new logic and transformation rules. The errors do not only occur when a batch job is running, but different errors can occur (and will) when developing a new batch job in DataStage. A simple mistake like a missing comma or similar can be hard to find when running a sequence job with the DataStage log only pointing to a specific transformer that could include over 100 different transformations. In lack of official support, developers can for some errors find a solution by using online public sources (forums, blogs, etc.) but the answers are limited compared to issues regarding SQL or any other programming language like Java for instance. Another drawback is that DataStage is not designed for team work. When a developer starts a new task, the development more or less has to be done completely by one developer. DataStage is simply not designed in a way that two developers could cooperate on the same project in an efficient way.

**Long-running batch jobs:** The amount of data collected by the company is constantly increasing in addition to adding new batch jobs added into production on a regular basis. This has also caused a problem and a broad network of job dependencies that cause bottlenecks during the data integration process when some jobs depend on other jobs that need to be finished before next process can start. This has created a scenario in which sometimes the data that should be updated during the night might be updated mid-day, resulting in old data being used for reports.

While a significant number of problems are encountered, replacing DataStage with another software, which would be an obvious solution, is at the moment not an option. IF Insurance is currently running thousands of unique batch jobs in DataStage every day. A scenario moving these jobs to a new software and re-creating the logic in a new software would require a lot of the company’s resources instead of focusing on new development. In addition, the software license for
DataStage is relatively inexpensive compared to other licenses provided by competitors, and the costs to replace the previous data integration tool. Many parts of the process are now being done outside DataStage with complimentary products such as SQL to speed up the process. But the need for DataStage could still be reduced by replacing some of the components in DataStage with a SQL solution.

1.3 Motivation

Developers face the previously described problems daily. The ETL process is vital for a business that uses data stored in the data warehouse and is used for most of the business processes. The data integration process is nothing that a large business like IF Insurance could manage without and therefore, improvements made will benefit the company also in the long run.

By analyzing the process, the aim is to identify parts of the ETL-process that could be improved to make the whole process more user-friendly and efficient. For instance, a small improvement, improving a part of the batch job that would save a couple of seconds in run time could play a big difference on a daily basis, if hundreds of batch jobs could save two seconds each in runtime. Making the process more user-friendly and less error prone would also save time and frustration among developers. A more efficient process will lead to each team being able to deliver more new developments in each sprint, while error solving will take less time. In addition, IF has recently acquired Tableau as a new business intelligence reporting tool, which makes reporting possible without any major coding knowledge.Currently IF has several thousands of Tableau users and the number of users increases every month, which of course creates a demand for new data.

1.4 Research Questions

This master’s thesis strives to answer the following research questions:
- What are the most relevant issues to be considered when designing an organizational data integration process?
- How can organizations improve traditional ETL-processing in terms of efficiency and user-friendliness?

1.5 Outline

The thesis starts off with a theory and literature chapter that will introduce the main topic of the thesis: data integration. This section will provide the necessary background information about the topic. The purpose of the literature review that is combined with the theory chapter, is to identify problem areas in the data integration process, and best practices to avoid pitfalls and shortcomings.

In the methodology section that follows the literature review the research method is presented. Then a data integration process is presented to analyze how a real-world scenario is performed in relation to the literature.

The final parts of the thesis will evaluate, based on the literature and case description section, how the data integration process done at IF Insurance can be improved. A discussion will follow this section with an unstructured interview with IF employees to reflect on whether the improvement suggestions really would benefit the company. Lastly suggestions for further research and limitations for the thesis are presented.
2 Theory and Literature Review

This section will provide the reader with needed theory, and a literature review about data integration.

2.1 Database Management and Design

2.1.1 Databases

Databases are one of the main components used during the data integration process and is where the data is stored. This section will give a brief overview of the database concepts to increase the awareness of different types of databases that can be found in different sources. The understanding of different types of databases is vital when one of the main tasks in data integration is to combine data from different databases.

Bhatia (2014, p.19) defines a database as follows; “A database is a collection of related files that are usually integrated, linked or cross-referenced to one another.” A database enables storage data and records that can easily be organized and accessed.

To make databases more user-friendly and to enable users to access and update data, a database management system (DBMS) is used upon the databases. A DBMS works as an interface that enables the user to access, store, update and retrieve data from a database. The DBMS manages the database schema that defines the logical structure of the database in addition to the database engine itself and finally the data.

Software is the main component of a DBMS, that consists of a set of programs to control, handle and manage the overall computerized database (Bhatia 2014). Statements developed in programming languages can request the DBMS to perform
operations like deleting, adding or updating data in the database. The other important components in a DBMS are:

Hardware – set of physical electronic devices like computers, disk drives, storage, etc. that is essential for a DBMS to work (Bhatia 2014).

Data – Bhatia (2014) states that data enables the main purpose of the DBMS that is data processing. Databases contain both operational data and metadata (data describing data).

### 2.1.1.1 Types of databases

Different types of databases are designed for different purposes. Some examples will briefly be discussed and presented in the next section.

Analytical databases (OLAP – On Line Analytical Processing) tend to be mainly static, read-only databases which are used for analysis (Bhatia 2014). Pathak (2007) highlights the response speed of an OLAP database as one of its main strengths. This type of database normally stores historical data, like sale transactions over the last five to ten years. Queries towards an OLAP database tends according to Pathak (2007), be very large, accessing several years of data at once.

Operational database – (OLTP – On Line Transaction Processing) – Bhatia (2014) mentions this database type is used to manage more dynamic bits of data compared to the OLAP database. This type enables the user to add, change and delete data from the database. The OLTP database are normally used to track real-time data such as stock quantities or how many items have been sold. Pathak (2007) mentions the OLTP database has not been designed for data warehousing. This is because this type of database is designed to handle lots of transaction meanwhile the data can be stored in hundreds of tables to avoid locking out its users. Pathak (2007) points out the data stored in this type of database tend to be inconsistent and historical data can be missing.
Relational database – Stores data in two dimensional tables that have rows and columns. This type of database is normally normalized which means the data is not repeated more often than necessary. Objects stored in a relational database are described in terms of string, integer and real-world data (Bhatia 2014). This means for example that customer data belonging to the same customer can be stored in several tables where all columns depend on a primary key. A primary key is a unique value in the table that identifies the column. There can also be a composite key, that consists of more than one column to uniquely define a row.

Database access language – used to perform any action with the database, like retrieving or updating data in the database (Bhatia 2014). A user submits commands to the DBMS and the systems execute to command to give the desired result. The most common access language is according to Bhatia (2014) SQL (Structured Query Language) and is also very relevant for upcoming parts in this master’s thesis and therefore, will be explained more thoroughly.

2.1.2 SQL

Bhatia (2014 p.182) defines SQL as follows; “SQL is a computer language for working with sets of facts and the relationships between them.” According to Pathak (2007) SQL processes large amount of data as a single unit. The same applies to multiple SQL queries run in a sequence are treated as one transaction.

As a result, SQL provides flexible transaction management, because when one SQL statement in the transaction fails, the whole transaction is aborted. Another benefit Pathak (2007) highlights is that SQL allows the use of constraints. Especially in data integration, constraints can be used to extract only the relevant data which cuts down the volumes of data processed and improves the performance of the data integration process over all. In addition, SQL syntax is fairly easy to read in
comparison with other computer languages when much of its syntax consists of English words.

A simple query to retrieve the first name of all employees whose last name is ‘Mc Greggor’ can be seen in figure 1

```
Select First_Name
From Employee
Where Last_Name = 'Mc Greggor'
```

*Figure 1, SQL example.*

The ‘Select’ syntax selects the ‘First_Name’ column while the ‘From’ defines which database and table to look in. The ‘Where’ syntax limits the result to only the first names whose last name is ‘Mc Greggor’. This is normally the introduction you see to SQL when reading literature, and it can seem like an easy programming language to learn. But SQL is just so much more than that, it enables the user to update data, create complicated transformations and calculation, joining tables and so on.

2.1.3 Stored Procedure

Stored procedures are widely used within IF for transforming and loading the data, this section will explain the concept.

Sunderic and Woodhead (2002, p.66) define stored procedures as follows: “Stored procedures are database objects that encapsulate collections of Transact-SQL statements on the server for later repetitive use. They are the equivalent of subroutines and functions in other programming languages “.

The composition of a stored procedure consists of a header and a body. The header defines and declares the name of the procedure in addition to output and input parameters and other miscellaneous processing options (Sunderic and Woodhead...
2002). An example of a stored procedure header can be seen in figure 2. The body of the stored procedure can contain one or multiple Transact SQL statements to be executed when the procedure is run. In addition, other procedures can be called within the same procedure and the result can be leveraged in the main procedure.

```
cREATE PROCEDURE NDP_LIB.WPExp_Transform
(
    IN pipeline_execution_id INTEGER,   --Parameter
    IN exomate_date --Parameter
) 
BEGIN
    /* Standard Logging Initialization */
    DECLARE program_database VARCHAR(128) CHARACTER SET UNICODE;
    DECLARE program_name VARCHAR(128) CHARACTER SET UNICODE;
    DECLARE start_tms TIMESTAMP(0);
    DECLARE diagnostics exception 1 error_message = message_text;
    CALL NDP_LIB.Write_Log(program_database, program_name, pipeline_id); -- Execute procedure within the same session
    BEGIN SIGNAL RESIGNAL;
    SET step_num = 3; SET start_tms = CURRENT_TIMESTAMP(6); SET processed_rows = 0;
    /* Procedure specific Logging Initialization */
    SET program_name = 'WPExp_Transform';
    SET pipeline_id = 'WPExp';

    Figure 2, Stored procedure.
```

2.1.4 Metadata

Metadata is used all over the data warehouse and is used for maintaining and managing data in the data warehouse. Nagabhushana (2006) states that metadata is one of the most important parts of the data warehouse. Metadata can according to Vidhya, Jeyaram and Ishwarya (2016) be divided in to two data types; technical meta data and business meta data. Technical metadata is used by data warehouse developers and administrators to be able to carry out management and development tasks. In addition, the authors mention that it stores data about data stores, data mapping, transformations, data access and history.

Business metadata’s purpose is to serve the users, helping them understand the content and find the data in the data warehouse. Therefore, business metadata is less structured and written in plain language unlike technical metadata. For instance, technical metadata can use cryptic language for a column name like
“ldv_Agrm_Id”, while business metadata for this column would be in plain language like “Individual_Agreement_Identity”. Nagabhushana (2006) complements business and technical metadata with optimizing metadata that includes aggregated definitions the front-end warehouse uses, and collection of query statistics that can help with tuning and optimizations of the data warehouse and its databases. To summarize, metadata is data describing data.

2.1.5 Data Warehousing

Data warehouses are built to provide the entire business with consistent data used for reporting and analytics. In addition, storing the data in a data warehouse removes some of the burden from the operational systems while it offers storage for historical data.

Reeve (2013 p. 37) defines data warehouses as follows:
“Data warehouses are data constructs (and associated applications) used as central repositories of data to provide consistent sources for analysis and reporting.”
The goal of a data warehouse according to Kimball and Ross (2013) is to provide the business users with easy understandable data that can be quickly accessed. The extract, transform and load process and other data integration technologies are responsible for moving the data through logical structural layers of the architecture into the data warehouse.

2.1.5.1 Data staging area

To be able to keep track of what data is moved into the data warehouse and help to solve errors found in the data in the data warehouse, the data coming into the data warehouse is usually stored in the original source format or is staged Reeve (2013). Kimball and Ross (2013) mention that staging of data is done in most ETL-tools, in this case, staging means writing data to a disk so it can be used later on in the ETL-process. Using a staging area also enables loose coupling with the timing when the
data was extracted from the source, and when it was loaded into the data warehouse.

The data in the data warehouse is no longer dependent on the structure of the data in the source system. The data is formatted in a consistent and logical way for the whole organization to use. Reeve (2013) states that the data warehouse also enables options to optimize the structure of the data to allow quick loading of high volumes of data from the various sources, this can also be done separately by storing data with these kinds of requirements in a data mart in the business intelligence layer, used for analysis.

Nagabhushana (2006) mentions that the data warehouse creates an integrated view for all sources in the organization. This enables organizations to see for instance the full scope and history of a customer, which maybe has not even been possible before when the data has been stored in separate sources. Connecting data between different sources is not an easy task and to keep track of what data is stored in the data warehouse, so called meta data is used.

Reeve (2013) points out the importance of the business, technical and operational metadata concerning for the data in the data warehouse. This includes a clear understanding of the meaning of the data, its origin, lineage and timestamps for events. The data provided in the data warehouse should also be accompanied with the associated metadata.

2.1.5.2 Data marts

Mourya and Gupta (2012) mention data marts can be seen as subparts of a data warehouse. According to Breslin (2004) a data mart is a database that can support one particular area of the business or a specific line of business such as sales. Multiple data marts can create a data warehouse. Depending on the source of the data, Mourya and Gupta (2012) categorize data marts into two types: dependent
and independent data marts. Dependent data marts are created by reading data from the data warehouse, meanwhile an independent data mart can be sourced from multiple operational systems or built from data generated by a department in the business.

2.1.5.3 Data warehouse architecture

There are many different data warehouse designs and architectures that organizations can implement when starting a data warehouse project. This thesis will give an overview of two commonly used architectural data warehouse designs, that will support the understanding of the main topic of data integration.

Ariyachandra and Watson (2006) have conducted a survey comparing the success of different architectural data warehouse designs to determine which design is the most successful. The most successful one turned out to be the bus architecture that will be introduced further below. In addition, the federated data warehouse architectural design will be explained to get a better overview of different architectural data warehouse designs.

Data warehouse architecture is according to Sen and Sinha (2005) a blueprint that enables planning, communication, maintenance, learning and re-use. In addition, it involves data design, technical design, and infrastructure design of hardware and software.

2.1.5.4 Federated data warehouse

A federated data warehouse provides the different functions within an organization with analytical capabilities. It is built from heterogeneous business intelligence
systems that aim to implement the key measures, metrics and dimensions of the business to provide the “Single version of the truth”. However, Jindal and Acharya (2004) point out a federated data warehouse does not aim to create a single platform that would carry out all the functional analysis. There are no data marts in this architectural design, all data marts are integrated physically or logically and as a result, all the data delivery comes from the centralized data warehouse (Ponniah 2011).

The federated data warehouse itself does not normally contain any data but generates consolidated answers to queries when multiple database systems cooperate (Samos, Saltor, Sistac and Bardes 1998).

Figure 3, federated data warehouse architecture based on illustration by Ponniah (2011).
2.1.5.5 **Bus data warehouse**

The bus data warehouse architecture is one architectural framework independent of technology and database platforms. The architectural framework guides the overall design and is divided into smaller business processes that can be implemented in realistic time frames. This allows developers to follow the same architecture while still working independently. Conformed dimensions and fact tables of any relational or OLAP-based dimensional model can be fully integrated into the bus data warehouse architecture (Kimball and Ross 2013). Ponniah (2011) mentions that this architectural data warehouse design is made up of multiple conformed data marts that can serve the whole business, not just a single department.

*Figure 4, Bus data warehouse architecture based on illustration by Ponniah (2011).*
2.1.5.6 Data quality

It is important that values in the data truthfully describes its real-world associated object. that is, if a home insurance was bought 01/03/2019 14:03:52 by the customer, the associated row for this insurance needs to have the same timestamp and product in the data warehouse. To achieve a good data quality Friedman (2006) recommends a holistic approach relying on people, processes and technology. In addition, the author mentions that organizations should constantly quantify and measure the data quality within the organization.

Data quality can be defined with the five C’s of data:

**Clean** – Making financial reports or press releases based on incorrect information can be very costly for a company. Dirty data can have missing values, invalid entries and other similar issues. Records representing the same real-world entity is common in every large database and are often referred as duplicates (Ribeiro, Cuzzocrea, Bezerra, and Do Nascimento 2016). These do not have to be identical copes, typographical errors, naming conventions and misspelling during data entry can be a reason for duplicates arising. Duplicates degrade the data quality and it is crucial to clean the data to ensure the quality of the delivered data.

Most source data contain dirty data to some degree, and therefore data profiling and data cleansing is a vital step when integrating data to a data warehouse (Sherman 2014). The data should according to Kimball and Caserta (2011) be unambiguous and values and descriptions should only have one meaning. For instance, a city name is not necessarily unique, there can be more than one city with the same name in a country. For a city name to become unambiguous, it can be combined with a zip code.

**Consistent** – The data should be formatted in a consistent way. Consistent data will result in the same or similar result no matter how the user handles the data. If the data would be consistently be stored with a certain time format, it would mean that there is no missing data for days or hours, depending on the format used. Kimball
and Caserta (2011) argue consistent data should only utilize one convention. For instance, the country Finland can be expressed in data like ‘FIN’, ‘FINLAND’ or even ‘SUOMI’, to ensure consistency in data, one should pick one of these conventions and stick to it in the data warehouse.

**Conformed** – The same data is used for decision-making by different units in the business and therefore, needs to be analyzed across common and shareable dimensions.

**Current** – How up to date the data is, in relation to the source system data, depends on the business. Making real-time or near real-time decisions, can be a must when dealing with credit card fraud detection and in that case, the data needs to be up to date (Sherman 2014). If the data will be used for reports summarizing the previous days sales result, a few hours of delay could be acceptable.

**Complete** - Kimball and Caserta (2011) state that there are two aspects of complete data. The first aspect can be met by defining columns to not be null in a table, it must be ensured that no null values can be inserted in to this column. As a result, all customers that should have a current address, do have one. To fulfill the second aspect of completeness, one needs to ensure there was no data loss during the information flow. This can be done by controlling the aggregated number of records.

### 2.1.5.7 Data governance

Data governance consists of multiple processes that ensure that the data assets in the company are formally managed. The processes warrant that data in the data warehouse can be trusted while keeping people in the organization accountable for any adverse event that could happen due to bad data quality (Sarsfield 2009). Newman and Logan (2006, p.3) define data governance as “the collection of
decision rights, processes, standards, policies and technologies required to manage, maintain and exploit information as an enterprise resource”

The main goals of data governance according to Brackett and Earley (2009) are to:

1. Define, communicate and approve data strategies, architecture, standards and procedures.
2. Track and enforce regulatory compliance and conformance to data policies and standards.
3. Sponsor and keep track of delivery of data management projects and services.
4. Manage and fix data related problems.
5. Give a good understanding of the value of the data assets and to promote it.

Soares (2015) presents an example in a case study where National Aeronautics and Space Administration (NASA) failed with its data governance which led to a loss of 328 million dollars back in 1999. Nasa’s engineers used wrong metrics in the data and used pounds instead of newtons, which caused a satellite to crash. This case study shows the result of poor data governance. Brackett and Earley (2009) state that data governance is most successful when it is an ongoing and continual improvement process.

2.1.5.8 Reference and Master data management

Within an organization, different areas, systems and processes need the same data. However, different business areas within the organization use the same information for different purposes.

For instance, sales and manufacturing departments could use the same data but they usually have different data quality expectations. Brackett and Earley (2009) mention that, organizations tend to create purpose-specific applications that
consist of similar data but with inconsistent data values and divergent formats. Inconsistency in data has normally a negative impact on the overall data quality within the organization. Master data represents the important items in an organization. This can be for example customers, products or employees. Reference data is according to Reeve (2013) a set of valid values associated with certain fields.

The reference data can be overlapping with the master data and can require frequent updates when reference data is attributes such as organizational hierarchy or industry codes that can change frequently. Reference data management controls defined domain values to classify and categorize data in the business. This process involves control over code values, standardized terms, business definition and relationships within and over domain value lists, (Brackett and Earley 2009). To ensure consistent and shared use of data across systems, master data management is used to enable the most timely, accurate and truthful version of a business entity. This is often referred as the “golden records” according to authors Talburt and Zhou (2015) and Brackett and Earley (2009) since it represents a single point of reference for a specific type of information. The golden records have to determine if there are two instances representing the same entity and which of them is most truthful.

Reeve (2013) states that both master and reference data are extremely important in data warehousing, because these are the types of data which users like to sort and display information with, and which data is consolidated for the whole business.

Changes to master data and reference data will be stored with start and end dates, i.e. validity periods when the value was valid and when the validity periods have ended or will end. Reeve (2013) mentions three ways to update master data: a change to the master data generates a new row of master data, the master data was always in its current and effective state or change the master data with its validity periods.
Brackett and Earley (2009) highlight three goals that reference, and master data management tries to accomplish:

1. To provide reference and master data of high quality in an authoritative and reconciled source.
2. To reduce cost and complexity by reusing and leveraging of existing standards.
3. To support data integration and business intelligence.

Issues with master data can cause major problems for an organization. If the master data is not well managed, an insurance company can for instance send marketing material for a home insurance product to a customer that already has a home insurance. This of course leads to wasted marketing funds, in addition to bothering the customer. The issue can originate in the business entity customer being differently defined in each system. Gordon (2013) mentions that information technology systems support different areas within an organization, and these are usually designed independently or off the shelf systems that can cause these inconsistencies in the data.

**2.1.5.9 Types of data**

Different types of data loaded into the data warehouse are treated differently and can have dissimilar life cycles. For example, some stored data only track changes in the source system and update the data accordingly in the data warehouse while some might extract data on a regular basis regardless of whether the data has changed or not (Reeve 2013).

Transactional data can consist of two types of transactions; business transactions and accounting transactions. Business transactions tend to have a life cycle with several events and statuses and can be managed in multiple ways. Usually the data warehouse keeps only one copy of the business transaction surrounding the event as well as when the transaction occurred, and when it was changed. Reeve (2013)
states that storing several ‘snapshots’ of the business transactional data in the data warehouse is rare, because analysis of the transactions is normally conducted on the event that caused the transaction, instead of analyzing the state of a specific transaction over time.

Accounting transactions are transactions that never change, and therefore, are only added to the data warehouse and never modified. Accounting transactions can be posted as debits and credits or offsets and are normally copied from operational applications (Reeve 2013). Each transaction is normally associated with master data in the data warehouse. Reconciliation is important according to Reeve (2013) in data warehousing. This involves periodic checks to confirm that the data sent from operational application systems was extracted and stored correctly in the data warehouse. Comparing financial totals and items sent/received should be done for each source and type of data.

2.2 Data Integration

If a company is using inconsistent data and of bad quality, the business most likely has some issues with its data integration process (Sherman 2014).

According to Lenzerini (2002) data integration is the problem of providing a unified view of data, stored in multiple different sources. Krishnan (2013 p.12) mentions the importance of having a unified view of the whole business: “Considering the agile requirements in decision making based on today’s fast-paced market and changing economic conditions, every business needs to have a 360° view of data across their organization.”

To achieve the goal of data integration the data resident in different sources needs to be transformed to provide the business with the five C’s of data, clean, consistent, conformed, current and comprehensive.

The data integration process, according to Sherman (2014), should be:
Holistic – Avoid inconsistencies and costly overlaps in the data.
Incremental – Practical and manageable.
Iterative – Analyze from previous projects and improve further development.
Reusable – To be able to assure the consistency of data, National (2010) points out reusing transformations that have already been created will increase both the efficiency and quality of the data integration process.
Documented – Can help in error solving and further development, where data is already defined and transformed.
Auditabe - necessary for government regulations and industry standards. General Data Protection Regulation (GDPR) is something every company handling personal or sensitive data needs to consider during the whole data integration process.

In addition, Dayal, Castellanos, Simitsis, and Wilkinson (2009) point out flexibility and scalability that complements Sherman’s (2014) view of data integration. Flexibility refers to the ability to take in previously unknown requirements or changing requirements as the years go by. Scalability is also important, when the data volumes increases, the ETL-process needs to be able to handle the growing amount of data to be processed.

Sherman (2014) states that when using productive and efficient methods, the data integration will be the most successful. Reusing data definitions and transformations is a smooth way to ensure data consistency. It is a productive way where one can get leverage from previous projects, but of course, this requires the data definitions and transformations to be already available.

Documenting data will assist in the reuse process. Documentation for all data integration processes should be created and maintained, containing source code, source and target tables. Sherman (2014) also mentions that a graphical visualization of the data integration workflow would be ideal. Data integration provides decision makers with fresh and accurate data that covers the whole organizations activity.
2.2.1 Data Integration Requirements.

To ensure that the data integration process runs smoothly and to enable it to provide information with qualities of the five C’s of data, there are some requirements before the data integration design can start.

Before extracting data from a source, it is important to know as much about a table’s schema as possible. Column names, data types, and keys are important metadata that is needed. Information about null values and minimum and maximum values can be a benefit.

It is also essential to know the data volumes; number of rows and how much storage space will be required. Like Peres, Rocha, Leitao and Barata (2018) state, data is constantly growing, which means an estimate for how much the data will grow in the near future, can be useful when planning where to store the data and to take into account expansion of the storage capabilities.

Historical data will also be a factor that affects the size of the data extracted, and statistics of how much historic data is stored in the source. If historical data is not needed, and only the freshest data from the source is needed, a lot of space in the data warehouse can be saved. Data loading cycles, that is, how often the data will be extracted from the source and updated in the target tables, is an important requirement.

Transformation logic needs to be documented before starting the data integration process. In this phase, the transformation logic applied is not the same as for analytical purposes before loading the data into a data mart (Sherman 2014). This requirement is to keep the consistency of the data in the data warehouse, by using same type of transformation logic. Both data and system dependencies need to be resolved. Data dependencies can be set by business rules or referential integrity constraints, or simply other tables that needs to be loaded prior to this table.
System dependencies or enterprise system interactions with the data integration process can be prerequisites (Sherman 2014)

2.2.2 Business Requirements

Gathering business requirements before starting with a data integration or business intelligence project is necessary to be able to fulfill the needs of the business. A data integration business requirement can be the velocity or latency of the data, i.e. how current the extracted data needs to be in relation to the data in the source system. Depending on the usage of the data, sometimes an update of the data in the data warehouse is enough, while sometimes the data needs be updated in real time or close to real time.

Sherman (2014) mentions historical and archiving requirements as an important business requirement. Determining whether history data should be stored or not, and if so, for how long should the data be stored. Data privacy and security issues are something the data integration process should consider. The data integration process transforms and delivers the data and therefore, needs to be adapted to the security and privacy requirements of the business. In addition, laws and regulations can also set requirements for what data can be stored and what needs to be anonymized. For instance, customer information and transactions can be illegal to store for a long period of time after the customer no longer conducts business with the company. Sherman (2014) mentions that many projects fall short due to poorly defined business requirements. Further he notes that a well created and defined requirements is the foundation for a successful solution. Ignorance among business users tends to be a major factor why some data integration projects are unsuccessful, and do not meet the business requirements. A solution for this is to assist business users when creating and defining requirements. Sherman (2014) questions how business users could ask for something they do not know is possible to have?
2.2.3 Data Profiling

When the potential data to be extracted has been identified from the high-level requirements, set by the business, data profiling is conducted to analyze and review the data to gain a better understanding of the format and content. Reeve (2013) states that data profiling is critical for a successful data integration process. The author also mentions that data profiling is essential for developing extracts that meet the business requirements. This statement is supported by Coleman (2012) who notes that data profiling could be regarded as a specific analysis that identifies features and characteristics of data sets.

Data profiling compares the format and content of the source in relation to business requirements, metadata and documentation. The analysis gives knowledge about data types, field lengths, column cardinalities, granularity, relationships between different sources, and other statistical data like percentages and distribution of values (Coleman 2012). The new insight gained from the data profiling analysis can detect differences in the source and what was anticipated from it. At the same time, it helps the business to avoid project delays caused by expectations not meeting the reality. According to Reeve (2013) there are tools available on the market that are designed for data profiling and that are capable of reporting descriptive data such as number of null values, distinct values, frequent values and even relationships between data across different sources.

Sometimes the data profiling stage can identify data quality issues in the source data, that leads to a business decision whether the correction of the problem will be done in the source data structure or if it will be moved to the data warehouse where it possibly could be corrected during the transformation step. Reeve (2013) mentions that the sooner one detects and uncovers a problem, the smaller the impact of it will be. Issues with source data can require a lot of time and resources to fix, and therefore, data profiling is an essential step to perform before extracting the data from a source.
Some problems and difficulties can slow down the data profiling step. The person performing data profiling will need access to the source data. The data stores can contain sensitive data such as health information which is strictly regulated who can access the data. Gaining access rights to a source can be a slow process but is essential to allow the person to perform data profiling on the data.

2.2.4 Extract Transform and Load

There are many kinds of methods for extracting, transforming and loading the data from production databases into the target database. Extract Transform and Load (ETL) is often referred as the process for performing this step, even though Extract Load Transform (ELT) or even Extract Load Transform load (ELTL) are used to some extent. This thesis will refer to this process by using the term “ETL” since it is the most frequently used in the literature. There are many tools available on the market that are more than capable of fulfilling the task of extracting, transforming and loading the data. It is also common that businesses design their own ETL-processes and tools to full fill their needs. Both approaches have their own pros and cons and will be evaluated later in this thesis. Designing and creating an ETL process can take up 70% of the effort in a data warehouse project and is therefore one of the key parts in data integration (Dayal, Castellanos, Simitsis, and Wilkinson. 2009).
2.2.4.1 ELT

Another approach much like the ETL-process is the ELT-process which stands for extract load and transform. The steps are the same as in the ETL-process but are performed in a different order. In the ELT process, the data is extracted from the source without any major transformations and dumped into the data base. The transformation process is then conducted with for instance SQL.

In the ETL-process, the tools do all the heavy lifting and complex transformations and only use the database servers for retrieving the source data and inserting the transformed data into the target table. One major drawback with the ETL-process is that the full potential of the database server is not used. The ELT process on the contrary uses the database to its fullest potential, and an advantage with this method is that little or no time is lost with data transmission. Another advantage Van (2012) mentions is that the database now holds both the transformed data and the original source data. Both methods have their own benefits and drawbacks and
every organization must choose the approach based on available resources and situation.

![Figure 6, simplified ELT example.](image)

2.2.4.2 Extract

The first step of the ETL-process is to extract the data from the data sources. The source can range from one table in a database, to several different tables in different types of databases. For instance, some of the customer data needed can be stored in a Postgres database, while same type of data can be gathered by another source into a db2 database. Both databases contain important customer data and therefore, need to be mapped together in the transform stage. One major challenge in the extract phase according to Patel, Patel, and Patel (2012) is that each data source has a unique set of characteristics that should be followed. El-Sappagh, Hendawi and El Bastawissy (2011) highlight two types of data extractions during the ETL-process: initial extraction and incremental extraction.

The first phase is normally done only once, when the data warehouse is built, and the ETL-process is designed. The incremental extraction updates the data warehouse with new data after the initial extraction is done. This type of extraction captures changed data in the source each time the ETL-process is run.

Before extracting data from a source, one needs to identify the unique identifiers and natural keys. Unique identifiers are columns that uniquely identify a row in a table (Kimball and Caserta 2011). A unique identifier could be the primary key of the table, but according to the author, this is not normally the case. A natural key is the
key businesses use to uniquely identify a row. From an ETL point of view an i.d. column could be the primary key of a table, while a status code column could be a natural key to use in addition to the primary key to uniquely identify a row (Kimball and Caserta 2011).

Understanding the relation between different tables before extracting data from a source is also vital (Kimball and Caserta 2011). Table relationships can normally be found in an entity relationship (ER) model that links keys and tables together in a database. Cardinalities for the tables can also normally be found in the ER-model and can be either one to one, one to many or many to many.

- One to one – The relationship is on the primary key of each table.
- One to many – Foreign keys pointing to the same primary key
- Many to many – Two tables have one table in common containing which has a compounded primary key with two foreign keys, one leading to the primary key of the first table and another one to the other table (Kimball and Caserta 2011).

If there is no ER-model created for the source, by re-engineering the database it is possible to create an ER-model. Kimball and Caserta (2011) also mention that a look-up table can be found in some source systems that contain static reference data for the whole database. A look-up table can contain a column that identifies which table and column certain rows can be found in.

Having a good overview of the data that is about to be extracted can significantly reduce any data quality errors in the future when the ER-model has been studied thoroughly. When knowing the key combination of the tables, one can easier optimize queries for faster extracts, that burdens the source system less.
2.2.4.3 Transform

Osborne (2008, p.1) defines a data transformation as “a mathematical modification of the variable to achieve a particular goal (e.g., normality, enhanced interpretability”. He also mentions that there are infinite varieties of possible data transformations in data integration, and these transformations can either be done during the ETL process or even in the reporting tool used. Transforming data on the go in an ETL tool can consume a lot of CPU, depending on the complexity of the transformation. In the transformation stage of data integration, it can according to Reeve (2013) be somewhere between extremely simple and nearly impossible to get the source data to be compatible with the target data structure.

Mapping and matching source schemas and target schemas is often a time-consuming task that requires a deep understanding of source and target data and its structure. This knowledge has usually been spread over numerous employees in the organization of whom some may not be working in the organization anymore (Doan, Halevy and Ives 2012). Schema matching is according to Sorrentino, Bergamaschi, Gawinecki and Po (2009) a major problem in data integration. This is because this step involves finding relationships between different schemas that can be very different from each other.

A simple transformation can be mapping a text or numeric value that is formatted consistently in both sources into the same target column. A difference in the technical implementation between the sources can require only a small adjustment in the format to get these values to match. A match between elements in different sources specifies how the elements correspond to one another (Doan, Halevy and Ives 2012)

String matching involves finding strings that represent the same entity, which is another difficult task. Manual insertions into data bases can be one reason to why some strings differ. Names for persons is just one example of what can create
difficulties matching two strings. For instance, in one source the name could be “John Doe” while in another source “John M. Doe”. There are many similarity algorithms Doan, Halevy and Ives (2012) suggest, such as Jaccard measure, and the edit distance measure. Although there are many tools that apply these algorithms, there is no tool that can perform a perfect string-matching task over a whole database (Doan, Halevy and Ives 2012).

Some values in the source data may only contain one set of possible values and might require a transformation to another set in the target. In some source systems, real world objects can be described by codes, consisting of numbers or letters. The transformation stage can use a lookup table that transforms the codes from one source system into a more user-friendly name. For instance, the code ‘1’ in the source could represent a real-world entity motorboat, while a code ‘2’ signify a sailboat. In a lookup or mapping table these codes and explanations would be on the same row, to enable an easy mapping to the target table, selecting the text value beside the corresponding code in the source system.

Quite often the transformation of data is way more complex than this. One record in the target table might consist of several records from the source system or the other way around. Missing or lacking data in the source might require some business or data quality rules to be applied.

### 2.2.4.4 Load

Loading step simply moves the transformed data into the target table in the data warehouse. The data can then be moved inside the data warehouse to different views used for different reports or mainframe systems. One of the main challenges when loading the data is according to Kimball and Caserta (2011) to handle the immense volume of data loaded at once. ETL tools offer the functionality to first update the target table and then insert, if new records are found. This method requires defined logic that separates existing records in a table from new records. A
drawback with this method is that it is extremely slow in most cases, especially when the data volumes increase. A so-called bulk load utility is another functionality that can be used, which will significantly improve the load performance in addition to decreasing the database overhead (Kimball and Caserta 2011). Though, an update function is normally not available when using the bulk load utility function.

Parallel loading is another method that Kimball and Caserta (2011) list. The data can for instance be partitioned into segments based on year, and in parallel be loaded from different files into the data warehouse. This method of loading can be especially useful when loading large volumes of data.

Physical updates in a table require a great deal of overhead in the database management system, and these kinds of updates should therefore be minimized (Kimball and Caserta 2011). Kimball and Caserta also mention that deleting all the records that should be updated and bulk loading the new data could in some cases be a much faster way to get the data loaded into the data warehouse. Thus, the ratio of rows in the table to the rows to be updated plays a big role, whether this is method is a good option or not. In addition, all source systems do not store historical data which would make this method inadequate.

El-Sappagh, Hendawi and El Bastawissy (2011) note that the data is normally loaded into so called dimension tables and fact tables. These table types will briefly be explained in the following section.

Dimension tables provide the measurements in the data warehouse and is also the context for the much bigger fact tables. Fact tables consists normally of transaction data, meanwhile dimension tables give reference data to the fact tables (Butt, Quadri and Zaman 2012). Kimball and Caserta (2011, p.161) highlight the importance of using dimension tables and fact tables in data warehousing:

“Although dimension tables are usually much smaller than fact tables, they are the heart and soul of the data warehouse because they provide entry points to data. We often say that a data warehouse is only as good as its dimensions.”
The primary key in a dimension should be a single (meaningless) integer, also known as a surrogate key. Kimball and Caserta (2011) point out that a surrogate key is a unique integer value generated by the ETL process, that no other entity can assign.

Using integers as primary keys will speed up the processing time when joining dimensions and fact tables together. In addition to the primary key, there should be a natural key. The natural key, unlike the surrogate key is not meaningless, and therefore this key represents one more field extracted from the source (Kimball and Caserta 2011). Finally, the primary key and the natural key can be accompanied by respective descriptive attributes from the source. These can be both numeric and textual.

Fact tables contain the business measurements, if a measurement exists, a row for this measurement can be created in the fact table. Kimball and Caserta (2011) defines a measurement as “an amount determined by observation with an instrument or a scale.” The fact table can be defined by the grain of the table, that is the lowest level of detail of the measurement event. The fact table contain foreign keys that can be linked to the dimensions. The foreign keys can be the surrogate key in the dimensions that can be joined together, to receive the description of the measurement stored in the dimension.

2.2.5 Manual Coding versus Tool-Based Data Integration

According to Sherman (2014) data integration consumes around 70% of the time and resources in a data warehouse project. He also mentions that using ETL tools for the ETL-process has been the best method for more than a decade. However, even large enterprises tend to manually code extracts for their data warehouse. This chapter will give the reader a brief overview of the advantages and disadvantages of manual coding and tool-based data integration.
The costs related to ETL tools are expensive and can be far out of reach for small to medium sized businesses (Sherman 2014). In addition to the license costs for the product, hardware and memory, expertise is needed to tune servers, networks and databases that results in a heavy ongoing expense that could even exceed the budget for larger enterprises. There are of course less expensive options on the market that will require a smaller investment, some tools have also moved more towards an ELT approach which will require less infrastructure investments compared to an ETL tool (Sherman 2014). Sherman argues that the learning curve for the top tier ETL products, can be steep, and therefore result in many businesses turning to hand coding instead. Generating an SQL code in a stored procedure can be an effective way at the start, but according to Sherman (2014) manual coding is seldom documented, which can create data silos and eventually end up as an enterprise nightmare. In addition, if hundreds of undocumented stored procedures or SQL scripts become the method to load the data warehouse, each change according to Sherman (2014) will consume an increasing amount of time and resources just to maintain the ETL-process, while the documentation seldom is updated.

ETL tools include prebuilt transformations and functions that are normally used in data integration processes. Hence many of these tools have inbuilt data cleansing, data conforming and data standardization processes to maintain data quality. Sherman (2014) states these steps can be too complex to be implemented by hand coded SQL. The documentation and generation of metadata is also generated to some extent by the ETL-tool, and it also offers restart and recover functionalities. One major drawback that Jun, Kai, Yu and Gang (2009) mention with an ETL-tool is the lack of flexibility.

Both methods have their advantages and disadvantages, and ultimately the resources and the needs of the business decide which approach will suit their needs the best. Sherman (2014) states that if a company would have the money to buy an ETL tool, why would it even consider spending time and resources on reinventing it?
Thus, a hand coded ETL can be designed specifically for a business, and if the
documentation is well done and updated hand coded ETL solution, could beat a
commercial ETL tool, even though Sherman (2014), finds using a commercial ETL
tool is better practice. Jun, Kai, Yu and Gang (2009) state that a combination of both
hand coding and the use of an ETL tool, could improve the efficiency and
development speed.

2.2.6 Testing

Testing is a vital part for the data integration process, not only will this step validate
the data quality and prevent errors in the future, but also according to Kimball and
Caserta (2011) it will excite the users. They also state that a successful ETL process
story will be spread throughout the organization and cause managers to practically
beg for their data to be brought in to the data warehouse.

Kimball and Caserta (2011) recommend a three-phased testing approach during the
data integration process to ensure good data quality. The first phase is a unit
testing that occurs during ETL development and right after, before moving on to
quality assurance testing (QA). The developer is responsible for this phase and their
task is to guarantee the data business required is delivered.

The second testing phase Kimball and Caserta (2011) suggest is QA testing. This part
can be done by another developer in a new environment much like the production
environment. This phase confirms that all ETL-processes are performing as
expected, i.e. following all business s rules and timeframe requirements. After this
phase, the ETL-Process is validated to work in the production environment.

The last testing phase Kimball and Caserta (2011) mention is the user acceptance
test (UAT). UAT enables the users of the data to have a practical demonstration of
the data before moving the entire ETL process to production. The business user can
confirm that the data meets the expectations and business requirements, and the ETL process is finally ready to enter production. Coleman (2012) argues that organizations should implement an automated data quality check.

2.2.7 Scheduling

To be able to execute batch operations, many organizations have standard scheduling tools available. ETL tools, and data transformation engines have both scheduling capabilities, to enable execution of custom code or ETL processes. Having all the jobs scheduled in the organizations standard scheduler that often runs at the night helps sorting out dependencies of the data coming from different source systems (Reeve 2013). Having all the scheduling done in one scheduler enables the business to have trained resources using the tool to follow up production and dependencies.

Running batch jobs during the night is a good way to use the capabilities of the systems you are paying for, also outside of working hours. In addition, extracting the data from source systems during nighttime reduces the burden and should not slow down the systems during the days when the business is uses them. The data will also be fresh and up to date in the morning when executives want an overview of sales results.

2.3 Summary of the Literature Review

This chapter will summarize the main points relevant to the problem description of this master’s thesis.

Table 1, literature review summary

<table>
<thead>
<tr>
<th>Reference</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagabhushana (2006)</td>
<td>Metadata is one of the most important parts in a data warehouse that can be used to optimize the performance.</td>
</tr>
<tr>
<td>Author and Year</td>
<td>Note</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>Sherman (2014)</td>
<td>Good data quality is vital when integrating data to a data warehouse. The data integration process should be holistic, incremental, iterative, documented, auditable and reusable. Data integration projects tend to fall short due to poorly defined business requirements. Graphical visualization of the data integration workflow would be ideal.</td>
</tr>
<tr>
<td>Sarsfield (2009)</td>
<td>Data governance keeps people within the organization accountable for data in the data warehouse, ensuring good data quality in the data warehouse.</td>
</tr>
<tr>
<td>Reeve (2013)</td>
<td>Data profiling is critical to ensure a successful data integration process. Reference and master data are extremely important in data warehousing. Controlling the data from the source correspond to the data stored in the data warehouse is recommended.</td>
</tr>
<tr>
<td>Dayal, Castellanos, Simitsis, and Wilkinson (2009)</td>
<td>About 70% of the effort and time in a data warehouse project is consumed by the ETL-process, which is a key part in data integration. Therefore, funding and the right resources are important to ensure this step is well executed. The data integration process needs to be flexible and scalable.</td>
</tr>
<tr>
<td>Patel, Patel, and Patel (2012)</td>
<td>Each source can have a unique set of characteristics that should be followed when extracting data.</td>
</tr>
<tr>
<td>Kimball and Caserta (2011)</td>
<td>Understanding of relationships between different tables is vital when working with data integration. Parallel loading can be useful when handling large amounts of data.</td>
</tr>
<tr>
<td>Doan, Halevy and Ives (2012)</td>
<td>Knowledge spread over multiple persons within the organization, can cause problems when a person no longer works in the organization.</td>
</tr>
<tr>
<td>Sorrentino, Bergamaschi,</td>
<td>Schema matching is one major challenge in data integration. It involves finding relationships between</td>
</tr>
<tr>
<td>Authors</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>Gawinecki, and Po (2009)</td>
<td>Database schemas that differ substantially from one another.</td>
</tr>
</tbody>
</table>
3 Research Methodology

In this chapter the method of choice is presented and motivated. In addition, the data and participants involved in the process will be presented.

Qualitative research is a commonly used research method to achieve a reliable result (Bryman and Bell 2013). It studies and analyzes data directly from field work observations, interviews or documents (Patton 2005). Holme and Solvang (1991) point out that the benefit of conducting a qualitative research is that it can resemble a normal conversation between two persons. Normally a qualitative research is conducted in a face to face meeting but can also be conducted via phone (Eriksson and Kovalainen 2008).

The quantitative research method approach usually involves a numerical or statistical approach while it is built upon an existing theory (Williams 2007). Quantitative research can be divided into three groups; Descriptive, experimental and causal comparative. The descriptive approach examines and identifies attributes of an object or situation in its current state. It can also include examining the correlation between different attributes. Experimental involves different kinds of experiments with variables and mathematical models that affect the outcome. Lastly the casual comparative approach examines what effect independent variables have on dependent variables.

Quantitative research is limited to study what can be quantified or measured meanwhile the qualitative research method tries to answer the research question by unquantifiable, personal, descriptive and social aspects of the object that is researched (Winter 2000).

The method of choice for this master’s thesis is action research. A qualitative study could have been an option, by interviewing employees about experiences to collect data. This would give the reader a much narrower insight of what the process looks
like and leave out technical details that could be improved and therefore also be used to answer the research question. And like Baskerville and Wood-Harper (1996) states: “Action research simultaneously assists in practical problem solving and expands scientific knowledge. This goal extends into two important process characteristics: First, there are highly interpretive assumptions being made about observation; second, the researcher intervenes in the problem setting”. The statement supports the choice of method, when the purpose of this master’s thesis is to identify issues within the data integration process, intervene and make it more user-friendly and efficient.

Action research is a common research method used within information systems research (Baskerville and Wood-Harper 1996). The action research methodology intends to have both action and research outcomes. It involves insight, reflection and personal involvement in the research field. Action research is also usually conducted in a real-world setting with people that are involved in the area that is investigated.

Action research is not a single large research method, but rather a class of research approaches. Despite different research approaches in action research, Baskerville and Wood-Harper (1996) highlight four common characteristics:

1. Change and action orientated.
2. Problem focused.
3. An ‘organic’ process that can include both systematic and iterative stages.
4. A collaboration amongst the participants in the research.

According to O’Brien (1998) there are four stages in the action research process that should be repeated; Plan, act, observe and reflect.

Planning and actions should be repeated until the effectiveness of the actions taken and adjustments are fully explored and implemented. Observations and reflections in action research can be for instance interviews, quantitative data collection or discussions.
Baskerville and Wood-Harper (1996) present a more simplified description of action research including only two stages. The first stage the author mentions is a diagnostic stage, that involves a collaborative analysis of the subject area where theories are formulated regarding the nature of the research field. The second step according to Baskerville and Wood-Harper (1996) is called the therapeutic stage. During this step collaborative change experiments are performed, and suggestions for change are introduced, and the results of these experiments are evaluated. One advantage with action research according to Dick (1993) is that the researcher can collect and record data while proceeding with the research. This helps the researcher to avoid huge amounts of data, that can for instance occur in qualitative research. In addition, the researcher can collect only the data that is relevant based on interpretations during the action research approach.

As an example, a published action research by Baskerville (1993) will briefly be presented to offer insight into a real-world case. The action research approach presented will also be relatable to this master’s thesis. In the problem case presented by Baskerville (1993), the case company had issues with completing a system analysis.

The observation phase in the action research performed, identified problems with data classes and large amount of data. In addition, the organization in turn, lacked expertise in database management.

The plan phase resulted in a plan to create a new prototype of a database design. The action taken in this research, was to acquire new hardware and software and implement a new database design.

The outcome of the action taken impressed the management and users in the organization. The new design provided by the action research proved to be a success, despite the company encountering some technical issues with implementing the new solution in form of a prototype.
This case study and approach to solve the current problem is similar to the problem presented in this master’s thesis. Therefore, action research is a suitable research method to answer the research questions for this master’s thesis.

The data used in this master’s thesis will be collected while performing a data integration process from start to finish at IF Insurance. Parts of the data integration process that are found difficult or less user friendly will be described in more detail to record more data that can be used to answer the research questions. In addition, the data collected in the literature review will be used to compare the data integration process done at IF to other research. The data that is presented in the thesis is anonymized and will therefore not violate any GDPR standards.
4 Case Description

Depending on the requirements the data integration process can vary considerably. Different source systems will require different approaches and sometimes collaboration between teams in the organization is required. The thesis will describe in detail the data integration process to include animals from a Danish source system into the data warehouse. The data that will be shown are test data and is not related to a real person, and therefore do not violate any GDPR standards.

The process begins with a business initiative; IF Insurance in Denmark is about to start selling dog insurances in a new source system that will be stored in a new database. To be able to perform analysis on the sales, and to be able to stay competitive on the animal insurance market in Denmark and maintain a good price level of the product the business decides to prioritize this task and forward it to the IT development teams. Animal data exist already in the data warehouse from source systems in Finland and Sweden. The business requirement is to create a Nordic solution to store animals in the same table in the data warehouse.

4.1 Business Requirements

The business requirements are set up in cooperation with an analyst who will be working with the data in the future. A cooperation defining the requirements will reduce the risk of shortcomings of the project. The major steps are summarized below to give the reader an overview of what needs to be delivered.

- Danish dog data from Microsoft SQL server into the Data warehouse
- Historical data need to be stored
- Updates of the data each morning
- One valid row for each insurance
- Use existing table (NDP.ANIMAL) as target (seen in figure 7)
- Map source data to target columns, add any new relevant column to the target table.
- General Data Protection Regulation (GDPR)

The target table with respective columns and definitions can be found in figure 7

```sql
Phys_Obj_Id INTEGER TITLE 'Physical object id' NOT NULL,
Ani_Vers_Id INTEGER TITLE 'Animal Version Id' NOT NULL GENERATED ALWAYS AS IDENTITY

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7, Ndp.Animal table definition.

4.2 Data Profiling.

Unlike the data warehouse that is managed with DBMS Teradata, the source for Danish animals is stored and managed with a Microsoft SQL server. Currently, there is no production data in the source, i.e. real data created from sales transactions and events, since the product is not yet sold from the new source system.
Though the business has populated the source with test data that resemble the real-world data, the volumes are quite small and can therefore leave out unknown data quality issues. This is a potential problem when developing the ETL process for the data when some unknown data quality issues can appear when the real data is provided. For instance, null handling or data type mismatches can become a problem. Unlike the target table in the data warehouse where the animal data is stored as one row for each animal, the attributes for animal in the source is stored in the same column as so-called information elements.

In figure 8 the exie_ex_Id column represents one dog, and this column works as a primary key that can uniquely identify a certain dog. Exie_Id and ie_code and respective code values are dog specific and can be used to filter out only dog related attributes in the source. The exie_value column is representing a real-world attribute depending on the exie_id and ie_code. For example when exie_id = 50000275 and exie_ex_Id = 36525889 the exie_value column value for this specific dog will be ‘Eddie’.

As we can see in figure 8, all attributes for the dog are stored in the same column, and it requires a look up table to understand what each value represents. In the data warehouse the users would like to see one row for each dog, hence each exie_value should be stored on the same row in own columns with more describing names. For instance, the exie_value for exie_id = 50000275 could be mapped to a column “Animal Name” where the value would be ‘Eddie’
Before continuing to the extracting phase, overall source system performance needs to be evaluated, at what time of the day the data can be extracted and in what volumes, and will the extract affect business users that are using the data in the source system.

If the source system can receive heavy queries without interfering with the overall business performance, performing extracts during business hours could be an option. After discussing the matter with business and colleagues with knowledge about the source, we decided that a nightly batch job would be enough to fulfill the business needs. Since there was no real-world data in the source, a well-executed data profiling step could not be conducted because of the lack of data. This could of course put the project at risk for not meeting the business requirements like Reeve (2013) states.

4.3 Extract.

When the data that is needed to fulfill the business requirements is defined, the data integration process can continue with extracting the data from the source. The extract process at IF has in recent years undergone major improvements in how the extract is done. A major part of the batch jobs in production is currently following
an ETL approach, with IBM InfoSphere DataStage doing the heavy lifting in extracting, transforming and loading the data.

Currently IF has moved more towards an ELT approach, while moving most of the processing to the powerful DBMS Teradata. DataStage is still responsible for extracting the data and to generate the metadata. The extract process is started by writing a custom SQL query that will extract the source data. DataStage has a feature that allows the user to write a custom SQL query, it also enables using timestamps as parameters. This speeds up the extract process as you can extract only the newest rows, and a full extract is not needed each time. This method can significantly reduce the volumes of data processed.

4.4 DataStage

When extracting new data from a source, IF Insurance uses so called pipelines that can contain multiple extracts done in a sequence. The animal data from the new source system is currently not being extracted, but related customer data is already being extracted in a pipeline named “WPExp”. The existing pipeline WPExp seen in figure 9 will be used when adding the Danish dogs to the data warehouse. The whole pipeline will not be explained as in detail in this master’s thesis, but the pipeline is shown in figure 9 to give the reader a good overall view of the process.
Each box in the sequence contains different kinds of extracts or meta data generating steps. The “JEL_WPExp_Waypoint_Exposure” component found in figure 9, will be used for extracting the animal data will be explained more thoroughly.

The first step is to add an open database connector (ODBC), “WP_Exposure” seen in figure 10. A custom created SQL query is developed to join tables in the source
system and extract the needed columns. The extracted data flows onwards in link L_in into a lookup component.

This lookup stage is used as a standard in all extracts done in DataStage at If. The WPExp Extract_Population data file connected to the lookup component contains a unique integer which will be inserted into the source table on Teradata, to identify the rows extracted in the same load. This integer is called “Extc_Pop_Id”, and will be used in a later phase where it is inserted in the target table. The process for generating this integer is currently completely done in DataStage, and the whole sequence job needs to be run before the file contains the new column Extc_Pop_Id. This is an outdated solution especially when If recently has moved towards an ELT approach. Since this part is used in all existing jobs, an improvement for conducting this stage could make an impact on the performance and usability of the Data integration process.

The Extc_Pop_Id column is generated when the sequence of jobs is run, seen in figure 11, located in the “NDP_Population_V3” box in the WPExp sequence job. In addition to the Extc_Pop_Id, a pipeline execution Id value is also generated, and the process for how to generate these values will be described next, in detail since this step has proven to be error prone and not so user-friendly.

Figure 11, metadata sequence job.

4.4.1 Pipeline Execution Id

When opening the first module in the sequence job, a parallel job seen in figure 12 can be observed. This parallel job simply inserts a new row into a table on Teradata.
The table generates a unique pipeline execution Id, and the timestamp when the row was inserted into the table. The transformer called “pipeline”, passes the pipeline name specified as a parameter into the table.

![Figure 12, Pipeline execution metadata generating step part 1.](image)

The second step in the sequence reads the Teradata table which was the final step in figure 12. The data moves on towards the transformer stage labeled “rename” found in figure 13. Table 2 provides an example of simple transformations that can be done in DataStage, to transform the data to desired format.

![Figure 13, Pipeline execution metadata generating step part 2.](image)

The data is then moved to files and to a Teradata table where the target column in table 1 can be found, and in this way storing the whole history of when each job has been run.

<table>
<thead>
<tr>
<th>L_in_srg_id.Ppln_Exec_Id</th>
<th>Ppln_Exec_Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>full_pipeline</td>
<td>Ppln_Id</td>
</tr>
<tr>
<td>pipeline_name</td>
<td>Ppln_nm</td>
</tr>
<tr>
<td>If IsValid(&quot;Date&quot;, emodateVar) Then StringToDate(emodateVar) Else SetNull()</td>
<td>Logl_Exec_Dt</td>
</tr>
</tbody>
</table>
4.4.2 Extraction Population Id

The purpose of the extract population generated in the next sequence is to generate metadata that connects rows in the target table which have been extracted from the source at the same time. When extracting data daily (sometimes several times a day) it can help significantly to fix any data quality issues in a later phase when millions of history rows can be stored in the target table, when a unique id identifies all the records extracted at the same time.

Figure 14, Extc_Pop_Id metadata generating step part 1.
The first component in figure 14 generates rows with dummy variables, that the transformer connected with L0 can use when creating a unique id for each source in the extract. The sources used by transformer “select_one_row_per_input” are manually inserted as parameters when designing a sequence job like WPExp. The transformer uses a stage variable to get all the sources inserted into the transformer:

If (@INROWNUM = 1 and TrimLeadingTrailing(source_1_path) <> '') Then TrimLeadingTrailing(source_1_path) Else If (@INROWNUM = 2... End as sourcePath.

Rest of the transformations are more straightforward and can be seen in table 3.
Table 3 transform - rename

<table>
<thead>
<tr>
<th>sourcePath</th>
<th>Src_Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline</td>
<td>Pipeline</td>
</tr>
<tr>
<td>@INROWNUM</td>
<td>Extract_Name</td>
</tr>
<tr>
<td>1</td>
<td>Join_Key</td>
</tr>
</tbody>
</table>

There can be up to 20 rows generated in this process that are copied to a table on Teradata and into a separate file.

![Figure 15, Extc_Pop_Id metadata generating step part 2.](image)

The last stage in the sequence for creating the Extc_Pop_Id column starts off with the lookup component reading of all the data that has been generated in the previous sequences. The “ppln_EP_W_Extract_Pop_Ids” component contributes with the source path that can for instance be the source database and table from where the extract is taken. In addition, this component provides an extract name, that in this case will be the number for the extract. The extract name was generated in the previous parallel job with the function “@INROWNUM” that returns the number of the row specified.

The “TC_Srg_Extract_Population_ID” component queries the database to get the generated Extc_Pop_Id from previous stage, and a parameter is used to get the right Extc_Pop_Id and extract name extracted from the table. The same parameter is then used to get the Execution Id from “ppln_Pipeline_Execution” data set. The values from these three sources are then joined together in the look-up stage. The
transformer stage is only renaming columns, before inserting the data into a database table and into a dataset. The dataset is used in the rest of the extract jobs to get the Extc_Pop_Id as seen back in figure 10, where the animal data was extracted.

The newly generated Extc_Pop_Id is combined with every row that has been extracted from the source and is inserted into a table named NDP_TMP.Src_WPExp_Exposure_IE_Extract. The data in the table corresponds exactly to the source data from where the extract was done. So far, no transformations have been done, except for adding the Extc_Pop_Id column to the dataset.

4.5 Transformation

All source data can now be found in Teradata platform where the transforming and loading of the data will be implemented.

<table>
<thead>
<tr>
<th>po_No</th>
<th>po_Id</th>
<th>ex_ObjectId</th>
<th>ex_Id</th>
<th>ie_code</th>
<th>exie_value</th>
<th>ie_Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000514</td>
<td>1</td>
<td>50000514</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000505</td>
<td>303</td>
<td>50000505</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000413</td>
<td>9</td>
<td>50000413</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000412</td>
<td>5</td>
<td>50000412</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000393</td>
<td>02</td>
<td>50000393</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000356</td>
<td>56-03</td>
<td>50000356</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000374</td>
<td>325</td>
<td>50000374</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000373</td>
<td>2019-02-14</td>
<td>50000373</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000260</td>
<td>0</td>
<td>50000260</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000279</td>
<td>1</td>
<td>50000279</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000276</td>
<td>0</td>
<td>50000276</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000276</td>
<td>1</td>
<td>50000276</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000275</td>
<td>Ede</td>
<td>50000275</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>SP001S... 9162348 1</td>
<td>37325612</td>
<td>IE-50000271</td>
<td>0</td>
<td>50000271</td>
<td></td>
</tr>
</tbody>
</table>

Figure 16, source data.

Figure 16 represents one dog in the source system, including keys that can be joined with customer data from other tables. The goal is to get one row for each animal in the target table. The “po_No” column represents the policy number that can be used to identify the customer owning the insurance. The ex_id in combination with
the policy number identifies uniquely an object that is insured under the same policy number. In this case shown in figure 16 where ex_id = 37326812, gives us 14 rows representing the same dog. Each ie_code has a value representing attributes like name and birthday for the dog. These attributes will be mapped to the corresponding column with a more user-friendly name.

The data shown in figure 16 has been extracted from the source and inserted into a table on Teradata, named NDP_TMP.Src_WPExp_Exposure_IE_Extract. The table contains the same data as in figure 16, but in addition to animals, information about cars and individuals. The transformation stage starts off by developing a query designed to fulfill the needs specified by business.

The transformations are done in a stored procedure that the developer creates. One major benefit with a stored procedure is that it can execute multiple SQL statements when called. As a result, it is possible to do transformations and insertions into different tables when calling a single procedure. Every stored procedure at IF must include two metadata generating stored procedures. When the main procedure is called to run, a procedure called write_log and write_log_success is also called. These procedures generate the metadata seen in table 4. The developer manually inserts the source and target tables used and the databases involved in the procedure. The rest of the data is generated by the write_log and write_log_success procedures.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>program_database</td>
<td>Name of the database the executed procedure is stored in</td>
<td>NDP_LIB</td>
</tr>
<tr>
<td>program_name</td>
<td>Name of the procedure (without database)</td>
<td>WPExp_Transform</td>
</tr>
<tr>
<td><strong>pipeline_id</strong></td>
<td>A short text identifier of the data integration pipeline. Can be used to find information and documentation related to the ETL process</td>
<td><strong>WPExp</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>pipeline_execution_id</strong></td>
<td>Numeric id identifying the specific (daily) execution of a pipeline. Used to connect with other metadata.</td>
<td><strong>1794581</strong></td>
</tr>
<tr>
<td><strong>start_tms/end_tms</strong></td>
<td>Execution start/end timestamp</td>
<td><strong>2019-01-19 18:34:01</strong></td>
</tr>
<tr>
<td><strong>last_step</strong></td>
<td>The name of the step the execution got to before ending, successfully or with error</td>
<td><strong>insert to target table</strong></td>
</tr>
<tr>
<td><strong>status</strong></td>
<td>Status of the procedure.</td>
<td><strong>success</strong>&lt;br&gt;<strong>fail - sql error</strong></td>
</tr>
<tr>
<td><strong>error_msg</strong></td>
<td>Possible error message</td>
<td><strong>sqlstate: Bad Character in format or data of X</strong></td>
</tr>
<tr>
<td><strong>teradata_user_id</strong></td>
<td>Teradata user id that was used to run this procedure</td>
<td><strong>USR_ID</strong></td>
</tr>
<tr>
<td><strong>sources/targets</strong></td>
<td>Semicolon-separated list of tables or views that are the sources of this procedure. Used for basic data lineage.</td>
<td><strong>NDP_TMP.Src_Wpe xp_Animal</strong></td>
</tr>
<tr>
<td><strong>processed_rows</strong></td>
<td>Total number of rows processed by this procedure.</td>
<td><strong>20 200 000</strong></td>
</tr>
<tr>
<td><strong>inserted_tms</strong></td>
<td>Timestamp when this log row was inserted to the log table.</td>
<td><strong>2019-01-19 18:33:58</strong></td>
</tr>
</tbody>
</table>
The stored procedures are called within DataStage which guarantees the procedure is executed at the right time when the extracts from the sources are done. The DataStage job in Figure 17 executes the procedure on Teradata, manually specified in “Exec_BTEQ” component.

![Figure 17, execute a stored procedure.](image)

The first step defined in the procedure with SQL is to remove any duplicates in the dataset, done with the code snippet seen in figure 18. The qualify row_number() syntax returns one row, based on unique values in the columns listed after the partition by. If there are more than one row with the same value in these columns, the code selects the first row of these ordered by column exie_Value.

```sql
-- de-duplicate information elements to only one per ex_id and ie_code, preferring the row with "minimal" exie_Value
select lue.po_No, lue.ex_objectId, lue.po_Id, lue.ex_id, lue.ie_code, trim(exie_Value) as exle_Value
from HDP_TMP.Src_HPExp_Exposure_IE_Extract lue
qualify row_number() over (partition by lue.po_No, lue.ex_objectId, lue.po_Id, lue.ex_id, lue.ie_code order by exie_Value) = 1
) unique_Its
```

![Figure 18, removing of duplicates.](image)

From the result of the subquery named “unique_IES” we can now start creating one row for each exie_value that represents a real-world attribute. In figure 19 a case function is used to only select the ie_codes related to dog insurances. If the ie_code equals the one specified between the single quotation marks, then the exie_value for this row is chosen. If the ie_code does not exist, we select null.

The exie_values returned from the case statement are first mapped to temporary columns named after the ie_code and the column description in the source system. The columns will still go through some minor transformations before mapped to the correct column name. In addition, maintaining the ie_codes and the name of the value description from the source system in the code, makes it easier to make changes in the future in case the ie_code changes and for other users to easily see
what column in the source has been mapped to the name for the animal in the data warehouse. The new columns are selected with the po_No, po_id, ex_id, ex_objectid in another subquery to be able to join the animal data with metadata coming from another extract extracted in the pipeline.

Data related to the animal rows in the ie_code subquery have been extracted into another table in Teradata and is called NDP_TMP.Src_WPExp_Exposure_Base. From this table we can connect the animal attributes generated in the figure 19. A left outer join on the columns po_No, po_id, ex_id, ex_objectid, from to ie subquery and the Src_WPExp_Exposure_Base table. A final transformation on the dog attributes is done where a mapping of the name still containing the ie_code from the source system are mapped to more user-friendly column names that can be found in the target table. In addition, a coalesce function is used, seen in figure 20, that will insert the value from the source system as long as it is not a null value. If the value is null the coalesce function will insert a ‘UNSPECD’ value instead. This will let the users know that this value for the specific column was unspecified in the source. Other column that are also not found in the source but are present in the target table is either replaced with a null value, ‘N/A’, or ‘UNSPECD’. These values could be automatically generated by the target table if you would give the column in the table a default value that will be used if no value is inserted.

The new animal attributes are connected to the insurance number (po_No) in addition to other keys that will be used when doing the final loading in to the target table. The almost ready transformed data is then inserted by a stored procedure into a table named NDP_TMP.LM_WPExp_Animal. The data is stored for approximately 24 hours in the table before it is deleted when new and fresh data is inserted. This enables the user to check the source data and the transformed data
in case of an error. This can help to identify issues during the loading process, like data quality problems. Though these issues need to be noted before the source data is deleted from the source or LM (Load model) table on Teradata. IF Insurance has recently improved the part of the data integration process that loads the data one final time into the target table in the data warehouse. The tool used is a company specific tool called ‘Lapio’ that will briefly be explained in the next section.

4.6 Loading

The final loading process has recently been improved at IF, and an in-house tool has been developed to do the final loading from the load model table into the target table. Since this part of the process recently has been improved, it is not necessary to outline in detail how the tool works, but instead give a brief overview of the final loading process.

Lapio is designed to batch load data into the data warehouse. A JSON configuration file specifies the load, while a windows command line application generates SQL queries based on the configuration and table metadata. The main functions in Lapio are:

- Bitemporal versioning, accepting two different types of input: delta and partial collection history.
- Surrogate i.d. mapping and generation.
- Checking the correctness of validity periods in the input data.
- Scoped update: only update the values in columns that are specified to be in scope for the update. Other column values are taken from existing rows when available.
- Maintaining the data warehouse metadata.

Lapio accesses the data from the load model table and maps the insurance number and object number to surrogate keys, Idv_Agrm_Agrm_Id, Ins_Obj_Grp_Agrm_Id.
These generated surrogate keys can be used across the data warehouse to quickly join tables to get information about an insurance plan. This enables the usage of indices that speeds up the processing time when handling significant amounts of data.

Before running Lapio, a new target table needs to be created. The new source system has additional valuable information that has resulted in new columns that need to be added to the target table. This step is done creating a backup of the old table and creating a new table with the old columns in addition to the new. Important here is to consider that all source systems do not have values to fill the new columns and as a result, need to be nullable or have a default value specified in the table definition. When the new target table is in place, Lapio can be run and shortly after the new data can be found in the target table. But before moving the jobs and data to production, the data quality needs to be improved by the business users. Figure 21 shows the result for some of the columns in the target table. Worth noting is that one row now represents one dog-insurance.

![Figure 21, Result sample.](image)

**4.7 Testing**

The business testing is normally performed by the person who has ordered the data into the data warehouse. The data is first manually tested by the developer who has created the solution. This step is done throughout the whole process and involves controlling of that the data is correct and corresponds to the source system. But before delivering the data to the business contact for business testing, a code review is done.

Two other developers from the unit goes through the SQL code, extracting and transforming the data to ensure that the code is understandable. This step is a
preventive action to avoid deploying unreadable code into production, while also looking at the solution created from another perspective to detect any flaws in the solution that could cause problems in the future.

The developer starts by manually controlling the validity periods in the target table, if there are more than one row for the key combination with more than one Vld_To_Tms = ‘9999-12-31 23.59.59’, then the data is not correct. The process continues with randomly picking a few insurances and using the insurance number in the graphic user interface (GUI), used by customer service when being in contact with customers. With the GUI it is possible to connect to the source and see all relevant data for each specific insurance. This ensures that the values in the data warehouse are correct in relation to the source. When the data looks fine the contact person in business is informed and given access to the data base on the test server.

4.8 Scheduling

The scheduling of the batch jobs is at IF Insurance a process which is outsourced to another company. Before moving any job that needs to be scheduled into production a ticket with the specific details of the batch job needs to be manually filled. The developer is responsible to figure out required dependencies and a scheduling time when the job should be run.

Creating a network of jobs with dependencies will ensure that each job is run in the right order, ensuring the consistency of the data. This is ensured by the DataStage logic seen in figure 22 runs a status check of the pipeline. The status of a pipeline can either be “S- Started”, “K- Unfinished” or “V – Finished”. Before a DataStage job can be initiated the part, which is shown in figure 22, the status for the pipeline must be “V – Finished” before updating the status to “S-Started”. If the job status is either “S” or “K”, it indicates the job has either failed or is still running. In addition
to the status, this part of the job also generates and stores information about job start time and end time. The timestamps generated by this step can later be used as parameters in a SQL query to only extract data from the source that has been entered into the source system since the last extract was done from the source.

*Figure 22, Run status of the pipeline.*
5 Findings and Discussion

This chapter will present the findings of the thesis. Firstly, the data integration process at IF will be compared with the literature, and what stands out. Afterwards the results of the thesis will be presented, firstly based on the literature, and secondly based on the data integration process performed at the case company.

5.1 Data Integration Process at IF Insurance in Relation to the Literature.

Setting up business requirements, testing and scheduling at IF Insurance is very similar to how the steps are outlined in the literature. Theory about data quality, data warehouse architecture was also applied at the case company. The biggest differences between the literature and the data integration process at IF, was within the ETL-process.

Most of the literature used in the literature review states that a tool designed for each step in the data integration process should be used. Most of these steps are either completely done with an in house developed tool, or with an in-house solution in cooperation with retail consumer programs at If Insurance. The ETL process at IF Insurance is more of an extract, load, transform, load (ELTL) kind of process that is not a method covered to as large an extent in academic research as the traditional ETL. Many authors remain skeptical to using hand coded tools and manually coded SQL extracts which is the main approach at IF Insurance.

5.2 Findings in the Literature

In the next section, the first research question will be addressed:
What are the most relevant issues to be considered when designing organizational data integration process?
To recall the data integration process is a combination both business and technical processes that combines data from various sources in to meaningful and valuable information that can be used for both analytics and decision making within the organization. When designing an organizational data integration process there are both technical and non-technical issues that need to be addressed and considered.

- Data warehouse architecture - Which type of architectural design will suit the business in question.
- Hardware - Where should the data be stored, on site or in a cloud-based solution.
- Software – invest in tools designed for data integration, ETL, data profiling and scheduling or develop an inhouse solution, with hand coded extracts?
- Resources – Available skills in the organization and the data integration team need to be evaluated when designing the data integration process. Available software, reporting tools, data sources need to be compatible with the data integration process and its tools. Funding is the most essential resource, as the literature also mentions, 70% of the time and resources in a data warehouse project is consumed by the data integration process. A well-funded data integration process will significantly reduce the risk of failure.

In addition to the technical aspects and resources that need to be considered is, according to the literature that the data integration process should be:

- Holistic – Avoid inconsistencies and costly overlaps in the data
- Incremental – Practical and manageable
- Iterative – Analyze from previous projects and improve further development
- Reusable – To be able to assure the consistency of data
- Documented – Gives leverage in error solving and further development
- Auditable – GDPR
- Flexible – Adjust to new and changing requirements.
- Scalable – Able to handle increasing volumes of data
5.2.1 What are the Most Relevant Issues to Be Considered when Designing an Organizational Data Integration Process?

The most relevant issues to be considered when designing an organizational data integration processes are data warehouse architecture, hardware, software and the organization’s resources. The data integration process should be designed to be: holistic, incremental, iterative, reusable, documented, auditable, flexible and scalable.

5.3 Case Company Analysis

The following part will use the data collected from the data integration process conducted at IF Insurance and attempt to answer the second research question of this master’s thesis:

How can organizations improve traditional ETL-processing in terms of efficiency and user-friendliness?

5.3.1 Extract Population Id and Pipeline Execution Id

The metadata generating steps described in chapter 4.4.1 and 4.4.2, can be completely replaced by similar logic created in a stored procedure on Teradata. Generating the metadata in DataStage has proven to be a solution prone to errors, both during the development phase and sometimes also in production. Since the data integration process at IF recently has started to move from an ETL approach towards and ELT process, it is sufficient to move and improve this stage outside the DataStage environment.

The template that is currently used when developing and creating new stored procedures can be updated to call two new procedures, NDP_LIB.Pipeline_Execution and NDP_LIB.Extract_Population. The user only needs
to manually write the source path into a predefined field. The newly created procedure will generate the needed metadata without the developer having to think about anything else.

NDP_LIB.Pipeline_Execution is the new procedure that generates the pipeline execution i.d.. The first step in the procedure is to generate a new surrogate i.d. for the pipeline. The surrogate i.d. is automatically generated by a table on Teradata, additionally a timestamp is generated when the procedure inserts the pipeline name into the target table. The pipeline name is specified in the main procedure that the developer has created, and it is passed to NDP_LIB.Pipeline_Execution procedure as a parameter. The pipeline execution surrogate i.d. is then read from the table and inserted into a table storing history metadata about when and what pipeline has been run. The target table for this procedure is the same as If has used before when storing the pipeline execution information, this part is now completely done outside the DataStage environment. The newly generated pipeline execution i.d. value is also passed forward as a parameter that other procedures can use within the same session.

NDP_LIB.Extract_Population is the second new procedure that can be used to generate Extc_Pop_Id column that has been previously described in this master’s thesis. Similar logic is applied in this procedure as in the pipeline execution id stored procedure. The first step in the procedure generates a surrogate id in the same way as for the execution id. The new generated Extc_Pop_Id is inserted into a history table, in addition to the source path specified in the main procedure and the pipeline execution id generated and passed forward from the NDP_LIB.Pipeline_Execution stored procedure. The NDP_LIB.Extract_Population requires the pipeline execution ID to be passed as a parameter but despite this, it enables some flexibility for the user. Some jobs in DataStage already have the pipeline execution generation step implemented, but when the developer update an old pipeline with a new extract of the Extc_Pop_Id it is necessary to have, to be able to use the tool Lapio to load the target table.
The developer can now simply choose to pass the Execution Id from DataStage to the new procedure. Another option is to remove the old pipeline execution generating step from the old job and generate both of these values within a stored procedure. Having two steps generating the same value is not an option. The generated Extc_Pop_Id value is passed forward as a parameter, the main procedure can use to insert into the LM table.

5.3.2 The New Solution from a Developer’s Point of View.

During the development phase of a new ETL Job, the developer needs to spend close to no time at all generating the metadata columns, Extc_Pop_Id and Ppln_Exec_Id. When the same step is done completely in DataStage, the developer needs to add several DataStage objects, that manually need to be filled. The Extc_Pop_Id that needed to be delivered in a file, required the whole sequence job to be run before being able to test a simple parallel job just because the Extc_Pop_Id was required. Additionally, the process was often error prone, if a small mistake was made during the implementation or during the testing phase, a lot of time was spent on trying to identify and correct the error, especially for new employees who are unfamiliar with the process.

An informal interview was held in one of the meeting rooms at IF Insurance. Four participants were invited to attend the session where the new proposed metadata generating solution was presented. Two of the attendees have over 20 years combined working experience in data integration, while the other two have less than five years of combined working experience. The session started with a demo, first showing the old solution and discussing issues related to it. One of the developers currently working partly in the production follow up team, mentions that the old solution for generating the metadata has been causing some batch jobs to fail recently, due to multiple jobs using the same resources at the same time. When the question of if any of the developers have faced problems during development of a new batch job including the old metadata generating steps was
brought up, one of the more experienced developers leans back in his chair, smiling while remembering all the struggles he had faced during his early years as a developer. Although he mentioned that nowadays he more seldom struggles adding the metadata parts to a batch job, thanks to his working experience. The other attendees agreed that there has been flaws in the old solution.

The new solution was presented for the first time for the attendees. First the source code of the two new procedures was described in detail, to give a technical understanding of the whole process. The second part of the demo showed the new solution in action was over at the blink of an eye. The audience was impressed by the simplicity and performance of the new solution compared to the old version. When asking the attendees whether they thought this solution could improve the data integration process done at IF, all of them agreed and pointed out that the new solution would improve the data integration process in terms of performance and spare time during new development.

To summarize, the new solution can save significant time during the development stage in addition to making the whole ETL process more user friendly and automated.

5.3.3 Performance

If we look beyond the improved usability and timesaving for the developer, performance improvements can also be noted. The old solution took on average 21 seconds to run while it created a bottle neck for the rest of the jobs in the sequence to wait for this job to finish. The new solution runs and finishes within a second and causes close to no overhead on the database.
5.3.4 Drawbacks with the Solution

The new solution has been roughly two months in production, and no major drawbacks have been identified. One disadvantage with the solution is that it is not suitable for all types of jobs. Some jobs do not require a stored procedure to load the data, and DataStage can even be easier and faster to use to load the target table. In these kinds of cases the new solution is not applicable, since it requires a stored procedure. Even though the solution comes with major improvements, it is not worth spending time on resources to update existing jobs, to use stored procedures to generate the needed metadata instead of the existing DataStage solution.

5.3.5 Code Repository

Many source systems contain similar columns to some extent, which should end up in the same column in the data warehouse. To ensure consistent data, reusing transformations is a good option. Source code for transformations done can currently be found in stored procedures, or in a Git Hub client that is a version control service where the source code is accessible.

Many transformations of different kinds of columns are already invented and can be found in one or more of the hundreds of stored procedures already created. Naturally, finding the right transformation can take time, especially if one does not know where to look for it. In the worst case, the transformation one is looking for has not even been created by anyone and one either spends hours writing one’s own code or tries to find something similar on Google. As National (2010) and Kimball and Caserta (2011) state, a repository with reusable transformations can significantly improve and speed up the data integration process.
IF Insurance could create a webservice where developers can enter the code snippet used for different kinds of transformations. The code could be categorized in groups to easily find what one is looking for. As an example, date, string operations, and social security number calculations are examples of groups that could be used. Dates can come in many different formats depending on the source system, but the target column in the data warehouse at IF is in most cases of format timestamp (0) and, as a result, the same transformations can be re-used.

5.3.6 Drawbacks and Challenges with Creating a Code Repository

A code repository has close to no disadvantages, code used from the repository has proven to be working in production and will go through another code review if the codes are used in another solution. However, to be able to reach the full potential of a code repository, it requires that the developers in the organization contribute. The more code and transformations that are found in the repository, the more its use will increase.

5.3.7 How Can Organizations Improve Traditional ETL-processing in Terms of Efficiency and User-friendliness?

Organizations can improve traditional ETL-processing in terms of efficiency and user-friendliness by implementing the new metadata generating solution presented in the thesis. The solution will make the process more efficient both in the development and production stages. The solution is executed very efficiently in less than one second, compared to the previous solution with an average runtime of 20 seconds. Implementing the new solution will also require close to no effort by the developer, which also makes the development phase more efficient. Therefore, the ETL-process will also be more user-friendly, especially for new developers, because the new solution is less complicated from a user perspective.
A code repository will also improve traditional ETL-processing in terms of efficiency and user-friendliness. Reusing code in a code repository will speed up the ETL-process development, when gaining leverage from already developed code. This makes the process more efficient, while it makes it more user-friendly for the developer, when usable and working code can be found in the code repository.
6 Final Remarks

This chapter will discuss the limitations of the thesis, and provide suggestions for further research.

6.1 Limitations of the Thesis

Data integration is a broad topic, and there is no best practice that could and should be used by all companies. Requirements and business need to determine the best approach in each case, not to mention that the data available in different organizations is never the same. This master’s thesis conducted a data integration process to identify improvements, but as mentioned, the way the data integration process is done varies from case to case and, therefore, limits the thesis to not being able to identify all possible improvements. There is plenty of literature available on the data integration topic, but the most recent focuses more on real-time data integration that is currently not a major business need for IF Insurance company.

6.2 Further Research

Business demands are changing as new technology emerges, so without a doubt there will be major changes on how the data integration process will be performed at IF Insurance company in the future. Further improvements could be investigated to replace the TIRW part and how stored procedures are currently called in DataStage. These parts could possibly be automated and require less effort during the development phase, if created outside the DataStage environment. Another topic to investigate more is how to keep documentation updated when new pipelines are deployed, and old pipelines updated. Maintaining and creating documentation is a vital part of the data integration process that this master’s
thesis only briefly addresses. Other interesting topics for further research could be the usage of data lakes and real-time data integration at IF Insurance.
7 Svensk Sammanfattning


Affärsintelligens (På engelska business intelligence) består av teknologier som samlar, analyserar och presenterar data för dess användare. Värdet på dessa verktyg, som används inom affärsintelligens, är mycket beroende av de tillgängliga data. Kvaliteten på data spelar en stor roll för att man ska kunna fatta korrekt beslut, dessutom kan data vara utspritt inom organisationen i olika databaser. För att möjliggöra analysering av all väsentlig data och förbättra dess kvalitet, utförs en dataintegrationsprocess som kombinerar data från de olika källorna och presenterar data i ett enkelt och förståeligt format för dess användare, i ett datalager (på engelska data warehouse).

Syftet med avhandlingen är att ge läsaren en inblick i ämnet dataintegration samt teori som kretsar kring ämnet. Ytterligare försöker avhandlingen besvara följande forskningsfrågor:

- Vad är det mest relevanta som bör beaktas då man designar en organisations dataintegrationsprocess?
- Hur kan organisationer förbättra den traditionella ETL-processen i form av effektivitet och användarvänlighet?

7.1 Teori

Databaserna är en av huvudkomponenterna inom dataintegration eftersom det är däri all information lagras. Ett databassystem hjälper användarna att få tillgång till data som finns i databaserna samt möjliggör funktionen att uppdatera, lägga till

7.1.1 Datalager.

Datalager är byggda för att tillföra företaget data av god kvalitet som kan användas inom analytik. Ytterligare minskar ett datalager bördan på de databaser som används för att operera den dagliga verksamheten. Data som finns i datalagret är inte beroende av formatet och strukturen i ursprungsdatabase, utan data har genomgått dataintegrationsprocessen som har format och förbättrat kvaliteten av data.


7.1.2 Extrahera, Förvandla, Ladda

Extrahera, förvandla, ladda (På engelska extract transform and load (ETL)) är stegen inom dataintegration som kombinerar data från olika källor. Det första steget går ut på att göra en extrahering ur det källor var de data som behövs är lagrat. Följande

7.1.3 Testning och Schemaläggning


7.2 Metod

är att datainsamlingen kan pågå under själva forskningen. Detta bidrar till att forskaren behöver handskas med mindre datamängder i jämförelse med de datamängder som kan uppstå vid kvalitativ forskning.

Informationen i denna avhandling kommer att samlas in vid ett utförande av en dataintegrations process gjord vid företaget IF skadeförsäkring. Målet är att finna problem i processen som kunde förbättras och därmed besvara forskningsfrågorna. Ytterligare används litteraturgenomgången som stöd för att besvara forskningsfrågorna.

7.3 Dataintegrationsprocessen hos IF Skadeförsäkring

7.4 Resultat och diskussion

I processen identifierades genereringen av metadata som ett problem som kunde lösas. Det tidigare sättet att generera metadatat på var en aning föråldrat då IF skadeförsäkring börjat styra bort från ETL-verktyget DataStage till att istället använda lagrade procedurer. Resultatet presenterar en ny lösning där hela metadata steget är gjort i två stycken lagrade procedurer. Dessa procedurer kan placeras i en modell som sedan varje utvecklare kan använda vid planering av en ny lagrad procedur. Detta resulterade i att det nya steget kräver ytterst liten ansträngning av utvecklaren, och metadata genereras nästan automatiskt. 


Andra faktorer som måste beaktas är att data integrationsprocessen bör vara väldokumenterad, återanvändbar, flexibel och skalbar då data mängderna växer.
8 References:


Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling*


Nagabhushana, S. (2006). *Data warehousing olap and data mining*


Patel, S., Patel, P., & Patel, S. (2012). *Overview of ETL process with its important*. Department of Computer Science, Fairleigh Dickinson University, NJ-07666, USA, Department of Computer Science, Rajasthan Technical University, India.


