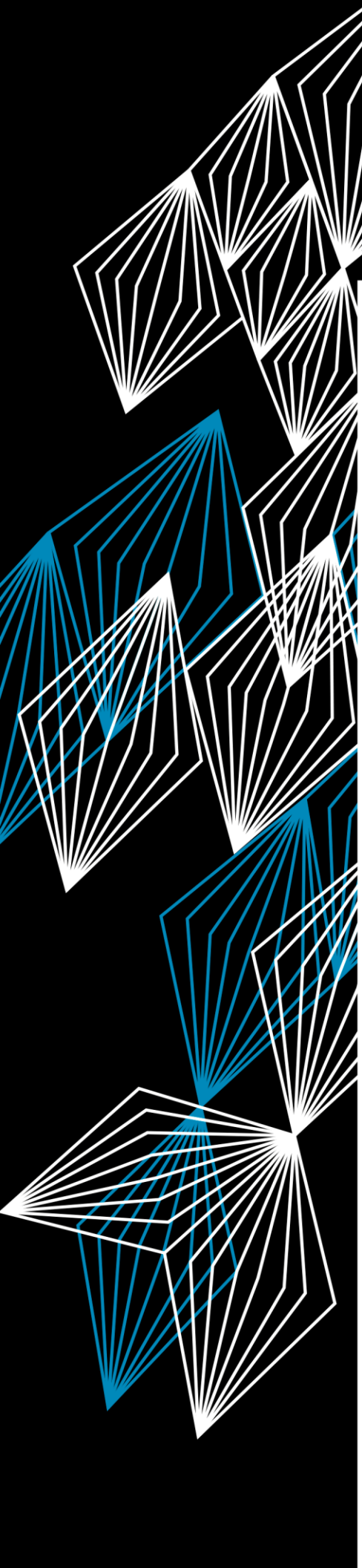


UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

Master Project

Choosing a Suitable Query Engine for Providing Observability and Accessibility for Dynamic Reporting of Business Data

July 2021

Student: Diana Ulici
Specialization: Business Information Technology

Examination Committee: prof. dr. M. E. Iacob
dr. F. A. Bukhsh

Company Supervisor: Samet Kaya

Executive Summary

Nowadays no one can imagine a thriving business without a proper data management and governance system for its applications data in place, in order to support the internal and external operations. Over the years, databases and their related DBMS (Database Management System) have grown in size, capability and performance, together with the development of cutting edge technologies and large scale business needs. Databases have become more fragmented and shifted towards being stored in the cloud, due to microservices and high tech advancements.

Choosing the suitable database suite is no easy task for the database specialists team, as the choice needs to take into account a multitude of criteria based on the organization's requirements, input from the business processes and the workflow of data, as well as the concerns regarding scalability, recovery and security. Furthermore, choosing a database suite is not sufficient anymore, as databases alone are not enough to make the information shareable and available everywhere it is needed. For this purpose, there should be a data management platform in place, that can handle all the steps from gathering, processing and making the information accessible. Therefore, a first step would be to analyze the structure of the information that will be held in the database, considering all the broad aspects of the business data, and then trying to shift towards one of the existing solutions available on the market, while keeping in mind the goals of the organization.

When it comes to an integration platform vendor like eMagiz, storing the data that flows through the system comes as an extra feature that can prove to give a strategic advantage in the business landscape. This comes of course with its challenges related to what kind of data can be stored, for what period of time, security concerns and so on.

For this Master Thesis, eMagiz, a Dutch IpaaS supplier, is the collaborative organization that provides the context for the developed prototype to be implemented and further validated. The prototype is mainly based on ElasticSearch as the chosen database storage option, and GraphQL as the querying engine. This solution aims to give the possibility of storing business data for a later use, while GraphQL provides the option of finding related data easily, while being able to visualize the data schema of the information stored.

LIST OF TABLES

LIST OF FIGURES

1. Introduction

For an organisation, one single source of data is critical for being able to easily access the information needed and gain insight from it. It also preserves data integrity, and gives fast and seamless access to it, without the need of connecting to different suppliers, which would require extra time, costs and effort. Without question, a company needs to have direct access and insight into its information resources in order to be able to survive in the business environment.

In general, there is a difference between “available” and “accessible” data that should be noted from the beginning. Even though there can be a lot of data available, that is seldom accessible. For example, some barriers that can interfere with it are the wrong formats or storage types, missing or ambiguous relations between various resources, character encoding or wrong headers. Fortunately, availability can be transformed into accessibility through different means and the right tools and knowledge. For example, formatting, cleaning and standardizing it to the wished format of the client would be one solution. [1]

The problem context revolving around the fulfilled background research is that, on one hand, organisations are uncertain of what database management platform to choose for their data storage, and what querying language would be the most suitable, as there are a multitude of options available, each coming with their pros and cons. On the other hand, there is a dire need for data governance along with data quality and data processing that could help in obtaining a proper data catalog for the organisation. Therefore, a dual research is needed to determine an overview of the existing data management architecture possibilities and fill in the gap, and that is broadly discussed in chapter 2 of this paper.

One might argue that the background research goal is too broad, but the purpose is to get a better understanding of data management platforms and querying in general, and then further on deep diving into more specific technology stacks available such as ElasticSearch and GraphQL, which show a lot of potential, are fairly new and trendy among tech savvy enthusiasts. Therefore, the literature research discusses the approach of NoSQL or Non-relational databases, along with examples, categories, different studies and comparison surveys. Additionally, a comparison between relational and non-relational databases is presented for an enhanced illustration of the differences. Further on, the second part of the research comes in, where data quality and data governance are discussed, which give more insight into the business facet.

1.1 Context

The company where the research and implementation take place is a Dutch iPaaS supplier, eMagiz, that offers a platform with which applications and systems can easily be integrated digitally. The company is active in the application and data integration market. The founding of the company can be traced back to the concept of Digital Transformation. One of the most important aspects of Digital Transformation is that organizations must be flexible in terms of integrations in order to adapt to quickly changing circumstances [50].

The often long running and phased projects can be managed from the online platform in terms of data integration. The flexibility offered allows to implement multiple integration patterns, including data & event streaming for high data volumes and messaging for processing individual

messages. With support for many protocols and formats, any integration can be built and data can be used effectively. The platform offers a detailed overview of the integration landscape using the integration lifecycle. The company offers a breadth of features, establishing itself as an Enterprise iPaaS.

An Integration Platform as a Service (iPaaS) offers customers the possibility to develop integration processes for data, applications and API's, whereby end-points can be located both on-premises and in the Cloud. The realization of integrations, deployment over environments and management of the landscape is all done within the platform. A platform that is aimed at Enterprise level, with which complex and demanding projects in terms of security, high-availability and disaster recovery can be realized.

The company makes use of a low-code platform as well. Integrations are modeled in a visual manner, as opposed to traditional programming. This makes integration development accessible and understandable, allowing the client organization to bridge the gap between business and IT. Using the "Integration Lifecycle Management" concept, all specifications and definitions are administered within the platform, allowing for quick, efficient and model driven development. This offers organizations the opportunity to involve business owners and stimulate collaboration with developers who do not require a background in computer science/programming more intensively.

This company description should give a better insight into the context of the existing platform that is used, because the prototype developed within the scope of this master project is partially integrated in the platform managed by the collaborative company.

1.2 Problem Statement

As described in the context section above, the company providing the IPaaS platform is already transporting the data for its customers, which means that after the transfer is finished, that specific information becomes irretrievable. The proposal of this master project is to create the possibility of saving the data while it is being transported and making it available for customers to use. For example, giving the option of visualizing the information and its schema, performing queries on it, and finding related data easily with the help of GraphQL.

When the information comes through, it could be stored in the database and later enabling an overview or calculating different statistics based on it. Factors that need to be taken into consideration include the type of data that is saved, the different formats that it comes through, for what period of time it will be desired to have it stored. Consequently, this function depends on the needs of the client and the use case that will follow, whether it will be for master data management, archiving, reporting, fraud detection or something else.

The problem at hand comes more as an enhancement to the existing functionality of the platform offered by the organisation, an extra feature that could prove beneficial for certain users. Gaining more insight into the data is a nice-to-have function, and that could be achieved by storing the data with the use of ElasticSearch. Other technologies could have been chosen for this purpose, but after an extensive literature research on possible options for storage and querying, that is presented in chapter 2 of this paper, ElasticSearch and GraphQL were determined to be used as storage and querying engines.

First, the reason why ElasticSearch was chosen as the storage solution is due to its scalability. ElasticSearch automatically distributes shards of an index across the nodes of a cluster and controls that they are loaded equally. Therefore, if more data needs to be added in future, Elasticsearch is a good choice as it scales horizontally. Another determinative aspect is that Elasticsearch does not impose schema on the documents in indices. If a new document is added to an index and there is a new field in this document, Elasticsearch will automatically update the mapping, which makes it agile. While scalability and schema-free documents are common for NoSQL systems, the combination of all three (scalability, agility, and performance) in one system is what makes Elasticsearch stand out from other systems.

Second, GraphQL was chosen because it provides a complete and understandable description of the data in the API and it can be used on top of any storage engine. It retrieves many resources in a single request and these queries access not just the properties of one resource but also follow references between them, which makes the discovery of related data possible.

1.3 Goal

The focus of this master project is to provide data observability in the context of the Dutch IpaaS provider, for the data that flows within the integrations from different systems. In order for that to be possible, a first step needs to be taken in the direction of storing the client information that is transported in the database layer, so it is not lost after the transfer through the platform is finished. Then, the second step of making the data accessible comes into play. In the Literature Study, more options are analyzed and detailed, but the chosen technology stack is ElasticSearch for the storage purpose and GraphQL for querying of the data. Finally, a React web application is used for the User Interface that provides the user with the possibility to query the data, visualize the data schema and find related data.

There is a key trend for analytics and BI platforms, and a focus on being able to make a better use of the data available. According to the Gartner Hype Cycle for Analytics and Business Intelligence [43], visual data discovery has long been the hallmark of the analytics and BI space. The result of more users analyzing more data is a need for organizations to focus equally on challenges relating to governance, data literacy, and education about the appropriate and ethical use of data.

An embedded solution in the existing platform is desirable to prevent the need of toggling to another application or changing the context. This way, users will be able to explore the data with no obstacles ahead, by using the platform that they are already used to.

1.4 Research Questions

The research questions are described using the design problem template proposed by Wieringa et al. [44]. The template is defining the problem context, the artifact that is going to be designed, the goal and the requirements, along with the stakeholders involved. The template is presented below:

How *to <(re)design an artifact>*
that satisfies *<requirements>*
so that *<stakeholder goals can be achieved>*
in *<problem context>*?

Applying the template to the research at hand, the main research question is formulated:

"How to design an efficient data management retrieval and reporting solution for client data based on ElasticSearch and GraphQL?"

In order to be able to reach a solution (an artifact) for the main research question, there are a few sub questions that need to be answered beforehand:

1. "What kind of data management architecture can be used for retrieval and reporting of client data?"
2. "What are the needs of the business users regarding this data management architecture?"
3. "How to design a data management architecture based on ElasticSearch and GraphQL for retrieval and reporting of client data?"
4. "How to integrate the data management architecture in the business context of a Dutch organisation?"
5. "To what extent is the designed prototype contributing to the goals of the stakeholders?"

In the interest of showing where the research sub questions have been answered, table 1.1 provides the correlation with the chapters:

Research Question	**Chapter from Master Thesis**
RQ 1	Chapter 2 - Background information and Literature Research
RQ 2	Chapter 3 - Requirements
RQ 3	Chapter 4 - Architecture Specification Model
RQ 4	Chapter 5 - Implementation
RQ 5	Chapter 6 – Validation and Evaluation

Table 1.1 Reference to what chapter from the Master Thesis answers the Research Questions

1.5 Research Methodology

The research methodology pursued throughout this master thesis is the one of Design Science Methodology (DSM) for information systems and software engineering by Wieringa et al. [44]. Since by the end of this project, a prototype (artifact) is developed to address a problem in the real world, this is an outcome-oriented research. A design science project iterates over the activities of designing and investigating. The design task itself is decomposed into three tasks, namely, problem investigation, treatment design, and treatment validation. The design cycle is part of a larger cycle, in which the result of the design cycle - a validated treatment - is transferred to the real world, used, and evaluated.

Figure 1.1 presents the design cycle of the Design Science Methodology, with its various steps and iterations.

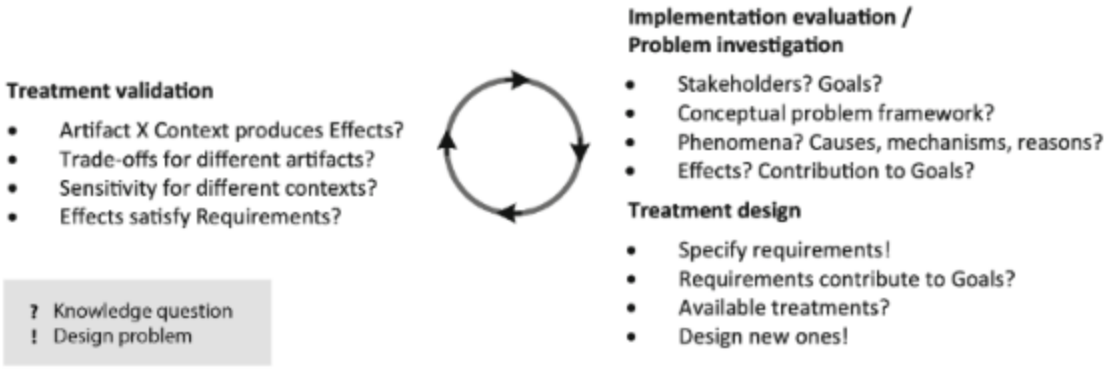


Figure 1.1 The design cycle of the Design Science Methodology

The consisting tasks, explained by Wieringa [44], are carried out in such a way that they answer the following questions:

- Problem investigation: What phenomena must be improved? Why?
- Treatment design: Design one or more artifacts that could treat the problem.
- Treatment validation: Would these designs treat the problem?
- Treatment implementation: Treat the problem with one of the designed artifacts.
- Implementation evaluation: How successful has the treatment been? This may be the start of a new iteration through the engineering cycle.

Starting with the problem investigation, the stakeholders involved are the Dutch IPaaS provider, its business clients as beneficiaries of the artifact and the author of this master thesis as the developer. The goals are broadly discussed in section 1.3 of this paper, and they revolve around providing a data management solution in the context of the already existing integration platform. During this phase, a literature research is conducted, in order to discover the available knowledge on the matter from published scientific resources.

The treatment design is the step where the artifact is developed, guided by the requirements available and taking into consideration the findings from the literature review.

In the treatment validation, the main question arises whether the treatment design would contribute to stakeholder goals if implemented. For this purpose, a single case mechanism experiment is done, to test the artifact prototype in the real world implementation.

The next step is the treatment implementation, that goes hand in hand with the treatment validation, and is based on carrying out the implementation within the given context.

The implementation evaluation is contained in the 5th sub question proposed and has the purpose of evaluating and validating the designed prototype.

1.6 Report Structure

This master project has the following structure: Chapter 2 gives the background information on the literature research conducted, introducing the problem context and presenting the findings. Chapter 3 expands on the requirements definition by using the means of expert surveys with open questions, after identifying the stakeholders and their goals. Chapter 4 provides an architecture specification model of the solution prototype proposed. Chapter 5 discusses the implementation in the context of the Dutch IPaaS provider as a case study. Chapter 6 shows the validation and evaluation of the design and finally, chapter 7 states the conclusions to the research questions and discusses the possible further work.

Table 1.2 shows the correspondence of the steps from the design cycle of the Design Science Methodology (DSM) by Wieringa [44] and the chapters of the Master Thesis:

Design cycle of the DSM by Wieringa	**Chapter from Master Thesis**
Problem investigation	Chapter 1 - Introduction Chapter 2 - Literature Study
Treatment design	Chapter 3 - Requirements Chapter 4 - Architecture Specification Model
Treatment validation	Chapter 3 - Requirements
Treatment implementation	Chapter 5 - Implementation
Implementation evaluation	Chapter 6 - Validation and Evaluation

Table 1.2 Correspondence between the steps from the design cycle by Wieringa [44] and the chapters of the Master Thesis

2. Literature Study

Chapter 2 focuses on gathering background information in the field of data management platforms and data governance and contains the answer to the first Research Question proposed, *"What kind of data management architecture can be used for retrieval and reporting of client data?"*. Additionally, it corresponds to the Problem Investigation step from the Design Cycle proposed by Wieringa [44], where the conceptual problem framework is established.

2.1 Research Method

The research method adopted for this background research is the Systematic Literature Review (SLR) to discover the previously done development regarding data management platforms and querying languages for data extraction. The methodology followed is based on Okoli et al. (2010) [8], providing a theoretical background for subsequent research, learning the breadth of research on the topic of interest and answering practical questions by understanding what existing research has to say on the matter. A rigorous stand-alone literature review must be systematic in following a methodological approach, explicit in explaining the procedures by which it was conducted, comprehensive in its scope of including all relevant material, and hence reproducible by others who would follow the same approach in reviewing the topic.

There are 8 steps present in the methodology, which are explained in Table 2.1:

Step	Name	Description
1	Purpose of the literature review	Clearly identify the purpose and intended goals of the review.
2	Protocol and training	The defined rules the reviewer will follow while systematically conducting the review.
3	Searching for the literature	Describe the details of the literature search and explain how the comprehensiveness of the search was assured.
4	Practical screen	Also known as screening for inclusion, this step requires that the reviewer be explicit about what studies were considered for review, and which ones were eliminated without further examination.
5	Quality appraisal	Also known as screening for exclusion, the reviewer needs to explicitly spell out the criteria for judging which articles are of insufficient quality to be included in the review synthesis.
6	Data extraction	Systematically extract the applicable information from each study.
7	Synthesis of studies	Also known as analysis, this step involves combining the facts extracted from the studies using appropriate techniques, whether quantitative, qualitative, or both.
8	Writing the review	The review needs to be reported in sufficient detail that the results can be reproduced.

Table 2.1 The 8 major steps for the Systematic Literature Review

The purpose of the literature review was stated right in the introduction. It comprises two parts, the first one researching the available scientific literature with regards to data management platforms and suitable querying languages, and then a more specific dive into the ElasticSearch and GraphQL solutions, while the second part is focused on data governance and data quality for an organisation.

The defined rules of the review process, as in what inclusion and exclusion criteria were used are described further in section 2.3.2. The resources for literature search are explained in section 2.3. The data extracting, along with the synthesis of studies is presented in the results of the study section.

2.2 Research Goals

In order to be able to investigate specific literature on a specific domain, a main research question should be determined, as well as sub questions, which will establish and guide the direction of the search process. Since the purpose of the study is to find related work previously done with regards to data management platforms and querying languages, along with data quality and data retrieval, the **main research question** is formulated as follows:

"What kind of data management architecture should be used for retrieval and dynamic reporting of client data?".

As for **sub questions**, the proposed ones are:

a) *"What are the advantages and disadvantages of the databases and querying languages in use?"*
b) *"What key points need to be considered when choosing a database and a query language for an organisation?"*

From these research questions, the dual character of the literature review is revealed: on one hand, the first part will tackle the data management platforms topics, and will consider different kinds of non-relational databases, query languages and then elaborate on ElasticSearch and GraphQL. The second part will revolve around the business side of data storage, more specifically data governance and data quality in order to be able to realize the dynamic reporting goal.

2.3 Scientific Resources Used

Google Scholar was the first tool used in order to reach scientific publications that are publicly available on the internet. It is a general search engine with a large coverage that provides scholarly articles and proved to be a good starting point for the research. I have made use of the filtering option, choosing resources that are publicly available and have been published starting from 2015 up to the present, so that the source is more relevant and up to date with the latest development.

The second utilized tool was the IEEE (https://ieeexplore.ieee.org), which delivers full text access to the world's highest quality technical literature in engineering and technology.

This search approach uses the classic systematic keyword search method, that is based upon usage of relevant terms in order to filter the online resources and is explained in the next section.

2.3.1 Search Query Keywords

For the first part, as the main concern of the literature research is based on what data management platforms and data storage solutions are available on the market, along with related querying languages and then focusing on more specific technologies such as ElasticSearch and GraphQL, the search query keywords used for discovering the helpful literature are presented in table 2.2. For the second part of the literature research, keywords like data management platform along with data quality, data processing, data catalog and data governance come in handy. They are presented in table 2.3.

Keyword	Name
1	Data management platform
2	Data hub
3	Non-relational databases
4	NoSQL
5	Query language
6	ElasticSearch
7	GraphQL

Table 2.2 Search query keywords used to conduct the first part of the literature research on the scientific databases

Keyword	Name
1	Data management architecture
2	Data quality
3	Data processing
4	Data catalog
5	Data governance
6	Observability
7	Accessibility

Table 2.3 Search query keywords used to conduct the second part of the literature research on the scientific databases

In order to make the research traceable, the search method for both scientific resources is provided below:

Google Scholar: for the first part of the research, there was a filter set for publications 2015-2021 and the following advanced search was used: *data management architecture nosql "non relational" ElasticSearch OR GraphQL "query language"*.

For the second part of the research, the same year filter remains in place (2015-2021) and the advanced search is: *data management architecture quality processing catalog observability OR accessibility "data governance"*. The complete process can be seen in figure 2.1.

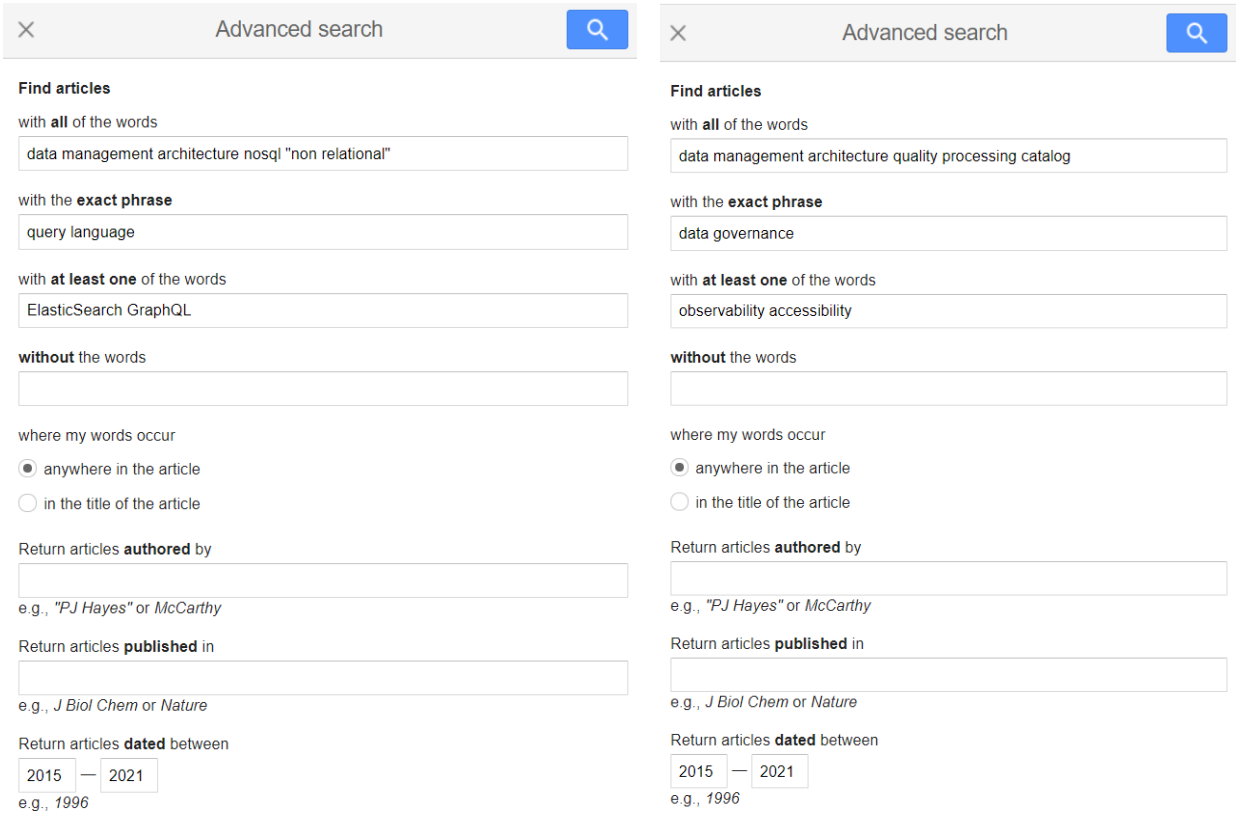


Figure 2.1 Advanced search on Google Scholar for the first part (left) and the second part (right) of the literature research

IEEE: for the first part of the research, set the filter for open access publications 2015-2021, and use the advanced search query:

("Full Text & Metadata":data)
AND
("Full Text & Metadata":management)
AND
("Full Text & Metadata":architecture)
AND
("Full Text & Metadata":nosql)
AND
("Full Text & Metadata":non-relational)
AND
("Full Text & Metadata":ElasticSearch)
OR
("Full Text & Metadata":GraphQL)

For both scientific resources used (Google Scholar and IEEE), the search for keywords is done in the full text and title of the articles, so a bigger area can be covered and more results can be gained.

2.3.2 Inclusion and Exclusion Criteria

Defining the selection criteria is crucial for identifying the relevant published literature and reducing the bias in the search process. Articles not meeting these specific inclusion criteria will not be considered for the literature review. Below, table 2.4 contains the inclusion and exclusion criteria used to aid in selecting the suitable resources:

Inclusion Criteria	**Exclusion Criteria**
Studies that are written in English and are peer reviewed	Articles that have no relevance to the keywords searched in their title or abstract
Studies that were published in Conferences or Journals	Studies unrelated to the main and sub questions of the literature review
Publication date is between 2015-2021	Articles that have duplicate content
Areas of studies are related to Computer Science, Engineering, Software, Computer Applications	Articles that cannot be accessed freely or with the UT institutional account

Table 2.4 Inclusion and Exclusion criteria for the literature review

2.3.3 Data Extraction

For the first part of the literature research, while using the above mentioned query search, the IEEE xplore yielded 9 results, while also having the filter applied for the published year between 2015-2021, among which 8 were Journals and 1 was from an Early Access Article. The advanced search from Google Scholar by using the search query mentioned above yielded 183 results, but it does not have a proper examination of the type of publication (for example, whether is a Conference paper or a Journal article).

For the second part of the literature research, only Google Scholar was used, on the grounds that it covers a multitude of resources, and it will yield duplicate results with IEEE.

In order to have a better understanding of the paper selection process, a flowchart is presented below with the steps taken in order to filter the relevant papers for both parts of the research:

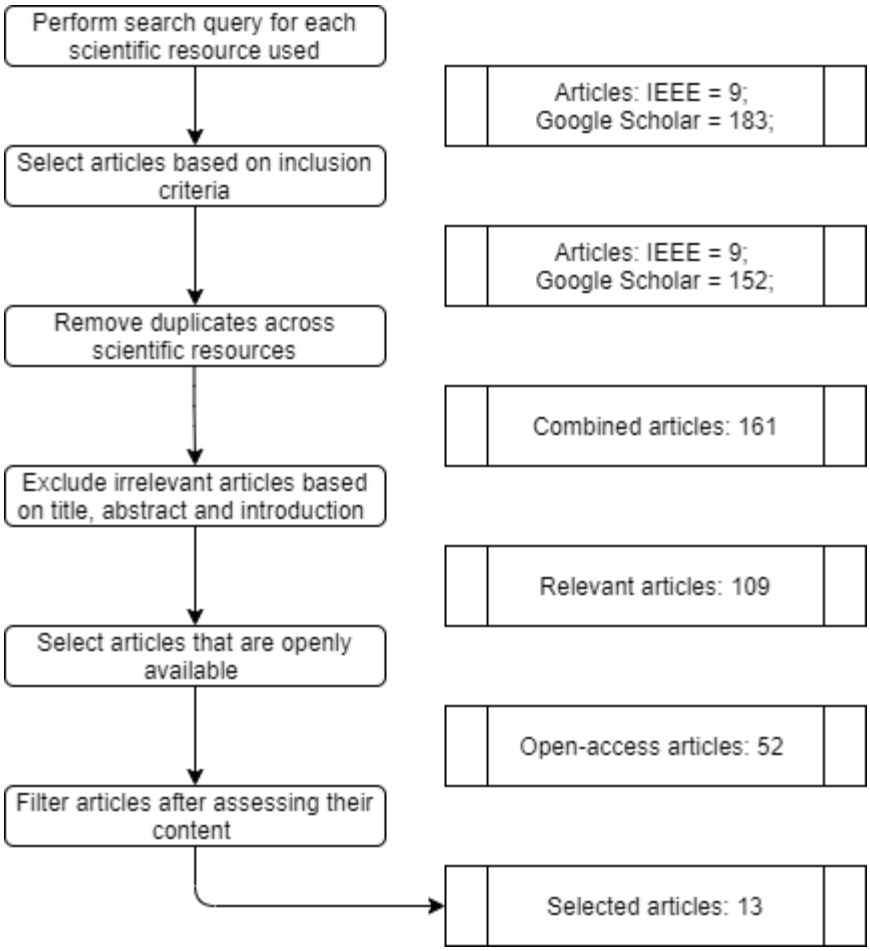


Figure 2.2 Process flow for selecting the relevant articles for the first part of the literature review

For the first part of the research, after the papers have been selected, each of them have been analysed on categorized on the basis of Table 2.5. Each paper is presented with a brief description, the title and authors, and it is shown why it was chosen as being relevant for the Data Management Architecture (DMA), what is the general perspective and whether it contains specific Definitions (D) for certain terms in the area that we are interested, Experiments (E) with hands-on implementations, a Comparison (CO) between the different solutions available on the market, or a Classification (CL) among existing types of data storage platforms.

For the second part of the research, the process flow is exactly the same, based on the same filtering criteria, with the difference that Google Scholar returned 1710 results in the first phase, and by the end of the filtering process, there were 8 articles selected that are further detailed in Table 2.6.

The next section will present the results with the extracted data, after the articles have been assessed by their relevance and the inclusion criteria stated before, for both parts of the literature research.

Findings from the first part of the literature research:

* *DMA- Data Management Architecture, D - Definition, E - Experiments, CO - Comparison, CL - Classification*

No	Reference	Title	Research Purpose	Perspective	DMA concepts			
					D	E	CO	CL
1	Čerešňák, Roman, and Michal Kvet.	"Comparison of query performance in relational a non-relation databases"	This paper deals with the database storage architectures, principles and differences. For the evaluation, relational databases are consecutively compared in the performance section to the non-relational oriented databases.	Implementation	x	x	x	x
2	Venkatraman, Sitalakshmi et al.	"SQL Versus NoSQL Movement with Big Data Analytics"	Two main revolutions in data management have occurred recently, namely Big Data analytics and NoSQL databases. The aim of this paper is to provide an understanding of their contexts and an in-depth study to compare the features of four main NoSQL data models that have evolved.	Overview	x		x	x
3	Vokorokos, Liberios, Matúš Uchnár, and Lubor Leščišin	"Performance optimization of applications based on non-relational databases"	This paper is focused on performance optimization of applications based on non-relational databases. Firstly there will be explained differences between relational and non-relational databases and also compared the types of non-relational databases. Then the advantages and disadvantages between MongoDB and ElasticSearch (SQL) database systems will be described and compared.	Implementation	x	x	x	x
4	Greca, Silvana, Anxhela Kosta, and Suela Maxhelaku	"Optimizing Data Retrieval by Using Mongodb with Elasticsearch"	The performance of organizations is directly dependent on the efficiency of data access, data processing, and data management. The purpose of this paper is to demonstrate the combination of using MongoDb with Elasticsearch for optimized data. This demonstration is focused on creating a "Data warehouse" for Agricultural products in Albania.	Implementation	x			x
5	Bhatt, Preeti	"Performance Comparison between Column	In this paper we describe the column-store NoSQL databases and the performance comparison between them.	Overview	x		x	

		Store NoSQL Databases”						
6	Khazaei, Hamzeh, et al.	"How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey."	We explore popular and commonly used NoSQL technologies and elaborate on their documentation, existing literature and performance evaluation. we will describe the background, characteristics, classification, data model and evaluation of NoSQL solutions that aim to provide the capabilities for big data analytics.	Overview	x	x	x	x
7	Mami, Mohamed Nadjib, et al.	“Squerall: Virtual Ontology-Based Access to Heterogeneous and Large Data Sources”	In this study, we present a unified architecture, which uses Semantic Web standards to query heterogeneous Big Data stored in a Data Lake in a unified manner.	Implementation	x	x	x	
8	Bondiombouy, Carlyna, and Patrick Valduriez	"Query processing in multistore systems: an overview."	In this paper, we give an overview of query processing in multistore systems. We start by introducing the recent cloud data management solutions and query processing in multidatabase systems. Then, we describe and analyze some representative multistore systems, based on their architecture, data model, query languages and query processing techniques.	Overview	x		x	x
9	Mehmood, Erum, and Tayyaba Anees	“Performance Analysis of Not Only SQL Semi-Stream Join Using MongoDB for Real-Time Data Warehousing”	Efficient stream processing for un-structured(NoSQL) and structured(SQL) data from various sources is required for the successful implementation of real-time data warehousing. We have done an analysis between un-structured and structured semi-stream join processing, using efficient database engine MongoDB at Extraction-Transformation-Loading phase.	Implementation	x	x	x	
10	Asaad, Chaimae, Karim Baïna, and Mounir Ghogho	“Nosql Databases:	In this paper, we present a survey of NoSQL databases and their classification by data model type. We also	Overview	x		x	x

		Yearning For Disambiguation”	conduct a benchmark in order to compare different NoSQL databases and distinguish their characteristics. Additionally, we present the major areas of ambiguity and confusion around NoSQL databases and their related concepts, and attempt to disambiguate them.					
11	Mami, Mohamed Nadjib, et al.	“The Query Translation Landscape: a Survey”	Methods that can simultaneously interpret different types of data available in different data structures and formats have been explored. Many query languages have been designed to enable users to interact with the data, from relational, to object-oriented, to hierarchical, to the multitude emerging NoSQL languages. Therefore, the interoperability issue could be solved not by enforcing physical data transformation, but by looking at techniques that are able to query heterogeneous sources using one uniform language. In this article, we survey more than forty query translation methods and tools for popular query languages, and classify them according to eight criteria. In particular, we study which query language is a most suitable candidate for that ’universal’ query language.	Overview	x	x	x	x
12	Mami, Mohamed Nadjib, et al.	“Querying Data Lakes using Spark and Presto”	We showcase Squerall’s ability to query five different data sources, including inter alia the popular Cassandra and MongoDB. In particular, we demonstrate how it can jointly query heterogeneous data sources, and how interested developers can easily extend it to support additional data sources.	Implementation	x	x		
13	Lu, Wen, Ligu Zhu, and Shufeng Duan	“Research and Implementation of Big Data System of Social Media”	Use the nosql database MongoDB to store the data and expand the management in time. Meanwhile, users can find important public opinion information from the historical database. ElasticSearch distributed full-text retrieval technology is adopted to quickly and accurately find relevant public opinion information.	Implementation	x	x		

Table 2.5 Selected articles for the first part of the study, relevant for data architecture platforms, querying languages, ElasticSearch and GraphQL

Findings from the second part of the literature research:

** DG - Data Governance, D - Definition, F - Framework, M - Methodology, E - Examples of implementations*

No	Reference	Title	Research Purpose	Perspective	DG concepts			
					D	F	M	E
1	Zhang, Ning, and Q. Yuan	"An overview of data governance."	We first review how the people understand the data, and various definitions of data governance, then examine the literature relating to framework, DQM, data lifecycle, privacy and security, compliance of data governance, next we explore the types of application areas of data governance based on the literature. The objective of this article is to provide an overview for data governance that can be used by researchers to focus on important data governance issues.	Overview	x	x	x	x
2	Priebe, Torsten, and Stefan Markus.	“Business Information Modeling: A Methodology for Data-Intensive Projects, Data Science and Big Data Governance”	This paper discusses an integrated methodology to structure and formalize business requirements in large data intensive projects, e.g. data warehouses implementations, turning them into precise and unambiguous data definitions suitable to facilitate harmonization and assignment of data governance responsibilities. We place a business information model in the center – used end-to-end from analysis, design, development, testing to data quality checks by data stewards.	Implementation	x		x	x
3	Foster, Kyle, et al	“Business Intelligence Competency Center: Improving Data and Decisions”	This article describes the development of a business intelligence competency center at a multi-line insurance company in the Midwest. It outlines the organization’s problems which led to the creation of the business intelligence competency center and the steps taken to ensure a successful implementation. Resulting from this experience is a set of best practices for business intelligence competency center implementation that, if followed, can lead to success for any company.	Use case	x			x

4	Attard, Judie, and Rob Brennan.	“Challenges in Value-Driven Data Governance”	This paper has the purpose of creating awareness and further understanding of challenges that result in untapped data value. We identify niches in related work, and through our experience with businesses who use data assets, we here analyse four main context-independent challenges that hinder entities from achieving the full benefits of using their data. This will aid in the advancement of the field of value-driven data governance and therefore directly affect data asset exploitation.	Use case	x			x
5	Nargesian, Fatemeh, et al	“Data Lake Management: Challenges and Opportunities”	The ubiquity of data lakes has created fascinating new challenges for data management research. In this tutorial, we review the state-of-the-art in data management for data lakes. We consider how data lakes are introducing new problems including dataset discovery and how they are changing the requirements for classic problems including data extraction, data cleaning, data integration, data versioning, and meta-data management.	Overview	x			x
6	Dai, Wei, et al.	“Data Profiling Technology of Data Governance Regarding Big Data: Review and Rethinking”	Data profiling technology is very valuable for data governance and data quality control because people need it to verify and review the quality of structured, semi-structured, and unstructured data. In this paper, we first review relevant works and discuss their definitions of data profiling.	Overview	x	x		x
7	Kachaoui, Jabrane, and Abdessamad Belangour	“From single architectural design to a reference conceptual meta-model: an intelligent data lake	This paper describes a new architecture implementation for DL systems with optimal management of metadata. This process treats data from heterogeneous data sources and with a combination of Data Warehouse (DW) for better management of structured data. This new approach offers companies an answer to data management problems and information availability.	Implementation	x	x		x

		for new data insights”						
8	DeStefano, R. J., Lixin Tao, and Keke Gai	"Improving data governance in large organizations through ontology and linked data."	In the past decade, the role of data has increased exponentially from something that is queried or reported on, to becoming a true corporate asset. While there are many tools and methods to increase a companies’ ability to govern data, this research is based on the premise that you can not govern what you do not know. This lack of awareness of the corporate data landscape impacts the ability to govern data, which in turn impacts overall data quality within organizations.	Overview	x	x		x

Table 2.6 Selected articles for the second part of the study, relevant for data governance, data quality and data accessibility and observability

2.4 Results for the First Part of the Literature Research

2.4.1 Non-Relational Databases

Non-relational databases do not use the Relational Database Management System principles (RDBMS) and data is not stored in tables, having a schema-less approach to data storage. A schema definition is not required before inserting data, and there is no need to adjust the schema if things evolve and change. [4] Among the advantages of Non-Relational databases it is worth mentioning the improvement of programmer productivity by using a database that better matches an application's needs and increasing data access performance via some combination of handling larger data volumes, reducing latency, and improving throughput. Another plus is that most NoSQL databases are open source and they can better deal with the increasing amount of complex data (big data) and distributed architecture.

A categorization of NoSQL databases contains the following 4 main types described by paper [5]:
1) *Key-Value databases* - which are the simplest NoSQL data stores to use, from an API perspective. The data consists of a key which is represented by a string and the actual data which is the value in the key-value pair. The data can be any primitive of programming language, which may be a string, an integer or an array or it can be an object. The most popular examples for such databases are Redis, Riak, etc. They are generally useful for storing session information, user profiles, preferences, shopping cart data. It should be avoided using Key-value databases when there is a need to query by data, having relationships between the data being stored or the need to operate on multiple keys at the same time.

2) *Document databases* - which store and retrieve documents as XML, JSON, BSON and so on. The most popular document database is MongoDB, which provides a rich query language. They are generally useful for content management systems, blogging platforms, web analytics, real-time analytics, e-commerce applications. It should be avoided using document databases for systems that need complex transactions spanning multiple operations or queries against varying aggregate structures.

3) *Column family stores* - these databases store data in column families as rows that have many columns associated with a row key. One of the most popular is Cassandra. They are useful for content management systems, blogging platforms, maintaining counters, expiring usage, heavy write volume such as log aggregation. It should be avoided using column family databases for systems that are in early development and have changing query patterns.

4) *Graph Databases* - schema-less databases which use graph data structures along with nodes, edges and certain properties to represent data. Nodes may represent entities like people, business or any other item similar to what objects represent in any programming language. Properties designate any pertinent information related to nodes. On the other hand; edges relate a node to another node or to some property. One can obtain some meaningful pattern or behavior

after studying the interconnection between all three vizited nodes, properties and edges. There are many graph databases, but the most popular ones are OrientDB, FlockDB, Neo4j, etc. They are very well suited to problem spaces where there is connected data, such as social networks, spatial data, routing information for goods and money, recommendation engines.

There are a few more types of non-relational databases that are brought up in the paper by Nishtha Jatana and al [7]. Therefore, continuing the classification from above:

5) Object Oriented Databases - commonly known as OODBMS, is a database system. It stores its data in the form of objects. This feature supports inheritance and hence reusability similar as in object oriented programming.

6) Grid and Cloud Databases - they make use of grid and cloud computing collectively. Grid computing is exploited to manage heterogeneous and geographically distributed databases while Cloud Computing provides easy access to remote hardware and storage resources.

7) XML Databases - a database management system that is used to store XML data. In all native XML databases, XML documents are the fundamental storage format. However, in some XML enabled databases, data of the XML document is divided into parts and these parts are stored within tables using another XML mapping layer. Some hybrid XML databases also exist which are a combination of native and XED's.

8) Multidimensional Databases - store the data as a n-dimensional matrix. All the useful aggregates are also precompiled and stored, allowing roll-ups and drill-downs to be answered interactively. Many products use this approach, like, Arbor Essbase [Arb] and IRI Express [IRI]. Multidimensional databases may use a relational database as a backend in which multidimensional queries are mapped onto equivalent relational queries. Products which use this concept are Redbrick and Microstrategy.

9) Multivalue Databases - earlier known as PICK database is such a database which understands three dimension data directly i.e. fields, values and subvalues. In this, value is a breakdown of field and subvalue is further breakdown of value. The tables in this database are extremely flexible such that if any of the changes are made in the database, there is no need to shutdown the database or rebuild the database. Also, multivalue databases have calculated columns as they can contain small programs for calculations.

10) Multimodel Databases - a blend of various other non-relational databases that provides a mix of advantages offered by various other types mentioned above.

Nonetheless, this is just one of the multitude of divisions that have been done for non-relational databases. The paper by Asaad [20] offers an extensive classification of NoSql databases as well, along with examples for each type, hoping to achieve a disambiguation among them.

After the classification of NoSql databases is discussed, the advantages and disadvantages of this approach are explained.

Starting with the benefits, one of the advantages of non-relational databases is their scalability and the fact that they provide superior performance while their data model addresses several issues that the relational model is not designed to address, for example large volumes of structured, semistructured and unstructured data, agile sprints, quick iteration, frequent code pushes, object-oriented programming, efficiency, monolithic architecture. [4]

Among disadvantages of non-relational databases, a few have been covered by paper [7]. Most of the Non-Relational databases are open source software and though well appreciated, it compromises in reliability as nobody is responsible in times of failures. Many of the Non-Relational databases are diskbased which implement buffer pool and multithreading hence require buffer management and locking features which add on to performance overhead. Many Non-relational databases provide BASE properties and sacrifice conventional ACID properties as a step to increase performance. This could mean that non-relational databases compromise on consistency within the database. Because of flouting ACID properties, the degree of reliability provided by non-relational databases is lower than what is provided by the relational databases. Developers have to rope in programming to apply ACID restraints which could have been provided easily in relational databases.

Diving deeply into specific types of NoSQL databases, there is a plethora of scientific literature that explains specific use cases. The next section will present these findings.

An analysis regarding the **column databases** (NoSQL) is presented in the paper by Bhatt, Preeti [15]. It shows what are the possible use cases for them:

- Content management systems;
- Blogging platforms;
- Systems that maintain counters;
- Services that have expiring usage;
- Systems that require heavy write requests (like log aggregators);

The same paper discusses the situations when this kind of column databases should be used: for semi-structured data, because it requires scalability and high performance; queries that involve only a few columns, aggregation queries against vast amounts of data and column-wise compression.

The situations when this type of database should be avoided are the following: for complex querying, if the querying patterns frequently change, if there is not an established database requirement, for incremental data loading, for Online Transaction Processing (OLTP) usage or if there are queries against only a few rows.

After the comparative analysis of various column-store databases, the study underlines that Cassandra and DynamoDB give high availability and partition tolerance but HBase, BigTable and HyperTable give consistency and partition tolerance. BigTable doesn't support concurrency control. Cassandra supports Master-Slave Architecture and HBase supports Hadoop Distributive Architecture. Today Facebook and other social networking websites prefer Cassandra over HBase because of its availability, open source, minimal administration, no SPoF (Single Point of Failure) and the fact that it provides security in every financial transaction. Companies like Bloomberg, Bank of America, Verizon and much more using HBas. HBase is good at intensive

reads, whereas Cassandra is good at writes. Cassandra lacks data consistency while HBase lacks data availability. Therefore, each database comes with its own advantages and disadvantages.

An innovative approach is presented in the paper by Mami [17], presenting semantic Web standards for heterogeneous data integration. For almost two decades, semantic technologies have been developed to facilitate the integration of heterogeneous data coming from multiple sources following the local-as-view paradigm. Local data schemata are mapped to global ontology terms, using mapping languages that have been standardized for a number of popular data representations, such as relational data, JSON, CSV or XML. Data of multiple sources and forms can then be accessed in a uniform manner by means of queries using a unique query language: SPARQL, employing terms from the ontology. Such data access, commonly referred to as OntologyBased Data Access (OBDA), can either be physical or virtual. In a physical data access, the whole data is exhaustively transformed into RDF, based on the mappings. In a virtual data access, data remains in its original format and form; it is only after the user issues a query that relevant data is retrieved, by mapping the terms from the query to the schemata of the data.

The pool of heterogeneous data residing in its original format and form is commonly referred to as Data Lake. It can contain databases (e.g. NoSQL stores) or scale-out file/block storage infrastructure (e.g. HDFS distributed file system). The goal is to facilitate accessing large amounts of original data stored in a Data Lake, i.e., without preprocessing or physical data transformation. Therefore, the result is building a virtual OBDA on top of the Data Lake, and calling the resulting concept a Semantic Data Lake.

An overview of three available NoSQL databases (Cassandra, MongoDB and Couchbase) is discussed in the paper by Băzăr, Cristina, and Cosmin Sebastian Iosif [6]. Cassandra is a distributed columnar key value database that uses the eventual consistency model. **Cassandra** is optimized for write operations and has no central node: data can be read from or written to any node in a cluster. It provides a continuous horizontal scalability and has
no single point of failure: if a node in a cluster fails, then another node comes and replaces it.
MongoDB is a NoSQL database, document-oriented, schema-free, which stores data in BSON format. A document based on a JSON, BSON is a binary format that allows quick and easy integration of data with certain types of applications. MongoDB also provides horizontal scalability and has no single point of failure. A MongoDB cluster is different from a Cassandra cluster or CouchBase cluster, because it includes an arbiter, a master node and multiple slave nodes. **Couchbase** is a NoSQL database, open source, document-oriented, designed for interactive web applications and mobile applications. Couchbase Server documents are stored as JSON. With integrated caching, Couchbase offers low latency read and write operations, providing linearly scalable throughput. Architecture has no single point of failure. The cluster is
easy to be scaled horizontally and live cluster topology changes are supported. This means that there is no application downtime when updating the database, the software or the hardware using rolling upgrades.

This paper also discusses some key criteria when choosing among the NoSQL databases described above, and it includes scalability, latency, performance, availability, and ease of deployment. Couchbase had the lowest latencies in the scenarios created for interactive applications because of cache objects built.

The extensive study accomplished by Khazaei, Hamzeh, et al. [16] aims to help users, individuals or organizations, to obtain a clear view of the strengths and weaknesses of well-known NoSQL data stores and select the right technology for their applications and use cases. NoSQL data stores can accommodate the large volume of data and users by partitioning the data in many storage nodes and virtual structures, thus overcoming infrastructure constraints and ensuring basic availability. Additionally, NoSQL data stores relax the transactional properties of user queries by abandoning the ACID system for the BASE (Basic availability, Soft state, Eventual consistency) system. In this paper, they survey the domain of NoSQL in order to present an overview of the relevant technologies to store and process big data. They present these technologies according to their features and characteristics. For each category it is shown a historical overview of how they came to be, a review of their key features and competitive characteristics, and a discussion about some of their most popular implementations. Big data has been forcing businesses to leverage new types of datastores that are more performant, economical, reliable and scalable compared to traditional RDBMS solutions. Selecting the right datastore is not a trivial task due to diversity and lack of standard benchmarks in this domain. There are research works and experiments to compare and contrast various solutions but none of them are truly generalizable and applicable for other interested parties. Therefore, this paper comes in to fill in the gap and present their findings and experiments.

2.4.2 Data Warehousing

Data warehousing is becoming increasingly useful for enterprises. Based on the specific needs of the organization, infrequently updated data warehouse environments do not support quicker business decisions and faster data recovery in case of transformation or load issue. Implementation of a real-time data warehouse provides a solution for this kind of problem. The paper by Mehmood [19] carries out a performance analysis to measure CPU and memory usage for real-time semi-stream join processing through two types of tests, unstructured and structured data streams using synthetic and real datasets. Their experimental results showed that under the given configuration, a fixed sized memory and constant CPU time is required for stream join processing for a given stream size, irrespective of the nature of data stream, which however increases for growing stream size. Stream processing with larger datasets need more disk I/O therefore increasing execution time and memory used.

For deriving intelligence out of data, the need of data warehouse (DW) is increasing nowadays. Rather than having multiple decision-support environments operating independently, which may lead to conflicting information, a DW unifies all sources of information. Decision support system architecture is composed of three important phases: DW building, exportation and Extracting, Transforming and Loading (ETL) processes which are responsible for extracting, transforming and loading data into a multidimensional DW. The basic purpose of ETL is to filter redundant data not required for analytical reports and to converge data for fast report generation.

Today's most common DW implementation is based on the relational model using SQL as its query language. However, Not Only SQL (NoSQL) DW solutions are being proposed by many researchers as they are more scalable and have better performance in comparison to relational databases. Typical RDBMS are inefficient for handling unstructured big data generated by Web, in contrast NoSQL has the capabilities of handling such kind of data.

2.4.3 ElasticSearch

One of the focuses of this literature research review was to gain more insight into previously done advancements regarding the ElasticSearch technology. This section will further discuss the findings.

Recently, new “big data” technologies and architectures have evolved to better support the needs of organizations analyzing data that is fast-moving and voluminous. In particular, **Elasticsearch,** a distributed full-text search engine, explicitly addresses issues of scalability, big data search, and performance that relational databases were simply never designed to support. In the paper by Kononenko, Oleksii, et al. "Mining modern repositories with elasticsearch." [9], a reflection upon its strengths and weaknesses is revealed, along with an in-depth explanation of what the capabilities of ElasticSearch are.

Elasticsearch is an open source full-text search engine written in Java that is designed to be distributive, scalable, and near real-time capable. While Elasticsearch and traditional RDBMSs differ in many ways, at the higher-level many of the core concepts of Elasticsearch have analogues in the RDBMS world. All data in Elasticsearch is stored in indices. An index in Elasticsearch is like a database in a RDBMS: it can store different types of documents, update them, and search for them. Each document in Elasticsearch is a JSON object, analogous to a row in a table in a RDBMS. A document consists of zero or more fields, where each field is either a primitive type or a more complex structure. A document has a Document type associated with it; however, all documents in Elasticsearch are schema-free, which means that two documents of the same type can have different sets of fields. Document type here is similar to the RDBMS notion of a table: it defines the set of fields that can be specified for a particular document.

Elasticsearch is based on Apache Lucene; each Elasticsearch index consists of one or more Lucene indices, called shards. The number of shards that each index has is a fixed value that is defined before the index can be created. When a document is added to an index, the Elasticsearch server defines the shard that will be responsible for storing and indexing that document. By doing this, Elasticsearch balances the loads between available shards and also improves overall performance, since all shards can be used simultaneously. While such automatic sharding is only one key part of the distributed nature of Elasticsearch, the other part of it is automatic distribution of shards among the nodes in a cluster.

Elasticsearch is a RESTful server, so the main way of communication with it is through its REST API. Communication between the Elasticsearch server and a client is straightforward. In the majority of cases, a client opens a connection and submits a request, which is a JSON object, and receives a response, which is also a JSON object. The simplicity of this mode of communication places no restrictions on the programming language used to implement clients or the platforms that they operate on; if a client can send HTTP requests, it can communicate with the Elasticsearch server.

Elasticsearch provides its own query language based on JSON called Query DSL. A given search can be performed in Elasticsearch in two ways: in a form of a query or in a form of a filter. The main difference between them is that a query calculates and assigns each returned document with the relevance score, while a filter does not. For this reason, searching via filters is faster than via queries. The official documentation recommends using queries only in two situations: for full text searches or when the relevance of each result in the search is important.

Regarding the strengths, first one to be mentioned by the paper is the Scalability. Elasticsearch automatically distributes shards of an index across the nodes of a cluster and controls that they are loaded equally. So if one expects to add more data in future and to accommodate the growth in data, Elasticsearch is a good choice as it scales horizontally.

The second positive aspect touched by the paper is agility. Data can be agile in terms of the number of updates or new records, in terms of constantly changing structure of a logical piece of information or document, or a combination of both of them. Relational databases are good at changing/adding data as long as the amount of data in a database is not too large. The time needed to perform a database maintenance (mainly recalculating indices) increases with its size. Elasticsearch can better handle agile data because of: a) each of the shards is being indexed/refreshed independently, and b) indices are constantly refreshed with fixed time interval, which means that it is unlikely that a shard has accumulated a lot of unrefreshed data. In the RDBMS world, a database schema is fixed and known before the first record arrives. If any updates might take place, the schema must be changed and this must be propagated to the records that are already in the database. If the database stores big data, this process can be very slow. Additionally, if the database is used in a domain where documents can have a lot of optional fields, the database can end up having large sparse tables that waste disk space for storing NULLs. Elasticsearch does not impose schema on the documents in indices. If a new document is added to an index and there is a new field in this document, Elasticsearch will automatically update the mapping. There is no need to change already stored documents since they do not have such a field. In addition, Elasticsearch can automatically alter the data type of a field if a value in a new document requires a “wider” type (e.g., changing integer to long).

The third and last advantage discussed is the performance. Best practises of the relational databases world dictate that each relational database must go through the normalization process during the design phase. By converting a database to a particular normal form bulk data is decoupled into several tables and the amount of redundant information is minimized. While the normalization benefits the create, update, and remove operations, it is likely to complicate the read operations. Most SELECT statements hit more than one table and they must be joined before the filtering conditions are applied. Although every RDBMS handles this operation as efficiently as it can, it is a time-consuming process if the query involves complex schemas. But since Elasticsearch is document-oriented it does not need to spend time on this preliminary step (i.e., gathering the data). Moreover, all shards within an index are searching for the documents satisfying some filter criteria concurrently, and after that results from all them are combined and returned. While scalability and schema-free documents are common for NoSQL systems, the combination of all three (scalability, agility, and performance) in one system is what makes Elasticsearch stand out from other systems.

When it comes to disadvantages, Security is the first one mentioned. By the time the paper was published (2014), ElasticSearch lacked security features such as authentication and access control, but nowadays it is not the case anymore.

The second and last disadvantage mentioned is the learning curve for the technology. The JSON origin of the Elasticsearch query language makes it really easy to start writing simple queries. However, query writing becomes more complicated if it involves nested objects. There is a special type of queries, nested queries, in Elasticsearch that must be used when one of the filter conditions is a condition on a field from a nested object. The use of such queries requires the

understanding of how particular documents are stored and analyzed by Elasticsearch (i.e., the mapping that is currently in use). Finally, Elasticsearch inherits some weaknesses of being a NoSQL system — lack of transactions, lack of JOIN operation, possible inconsistencies in data, etc.

Elasticsearch is best suited for the applications that are built to handle real time data that needs to be processed and analyzed in a rapid manner. Such applications include software analytics. Thousands of organizations worldwide including Netflix, Facebook, GitHub and Stack Overflow, have adopted Elasticsearch to help them overcome limitations of their old approaches in handling new demands of agile data processing and storage.

It quickly gained popularity and is one of the most favorite databases intended for searching. These main points sum up its main characteristics [13]:

- Default database setup suits most people even without any user adjustments.
- It is working in distributed mode by default.
- Peer to peer architecture without a single point of failure. Nodes are automatically connected to the distributed system for data interchange and state monitoring.
- Simple scalability of data capacity by adding nodes.
- Data lookup and analysis almost in real time.

Text analysis is a process which handles human language or complex data. ElasticSearch has tools which can dissect and edit words to make the search more effective. After adding text, it will split the text to appropriate expressions which are used in the invested index. The analyzer does three basic functions:

- Symbol filter – string of text is filtered to remove or substitute special symbols and HTML tags.
- Tokenizer – it splits the text into expressions and words based on spaces and periods.
- Token filter – it can change words based on current filter, for example uppercase or lowercase them, remove joins or add synonyms.

ElasticSearch has lots of in-build filters but allows to create custom ones and use them in the analyzer. The text is processed by the analyzer before adding into documents. The searched request is also processed by analyzer and only then is searched in the database.

Another paper that shows a successful implementation of the ElasticSearch full-text functionality is the one by Lu [29], in the context of social Big Data gathered from different messaging systems. With the rapid growth of the system data, the traditional search engine method has not been able to solve the index problem of massive data. With the increase of single index files, the search efficiency of index engines on index files is slower and slower. Distributed indexing technology can effectively solve such problems. The distributed index module of this paper is mainly based on ElasticSearch distributed search engine framework to design, Its main function is to set up the distributed inverted index for the mass data in the vertical field. Since the index method used in this paper is distributed index mode, each index data needs to be assigned to the corresponding index sharding according to the ID of the data. By default, ElasticSearch

adopts the shared-slice routing algorithm, which is the djb2 hash algorithm, which performs hashing calculation on the ID of the index document and the result of the hash calculation. When the index document is distributed, the index document needs to be analyzed in text. The sharding process is used to distribute the index steadily, and the text analysis process analyzes the index document and converts it into a Token flow. This article designs different text parsers for different types of index documents, including letter filtering programs, word segmentation programs, and word filtering programs.

In an inverted index file, an index object is a word in a document or a collection of documents. It is used to store the location of these words in a document or a set of documents and is the most commonly used index mechanism for the collection of documents. The key step in the search engine is to create an inverted sort index. The inverted sort index is generally expressed as a keyword, frequency, location, etc. It's equivalent to an index of the Internet's hundreds of billions of web pages, just like a book's catalog or label. The reader wants to see which theme is relevant to the topic, and can find the relevant page directly according to the directory. ElasticSearch uses an inverted sort file index structure.

2.4.4 Cloud Data Management Solutions

An overview of the **cloud data management solutions** is provided in the study by Bondiombouy [18]. A major trend in data management for the cloud is the understanding that there is “no one size fits all” solution. Thus, there has been a blooming of different cloud data management solutions, specialized for different kinds of data and tasks and able to perform orders of magnitude better than traditional relational DBMS (RDBMS). Examples of new data management technologies include distributed file systems (e.g. GFS and HDFS), NoSQL data stores (e.g. Dynamo, Bigtable, Hbase, Mongodb, Neo4j), and data processing frameworks (e.g. MapReduce, Spark). The problem of accessing heterogeneous data sources, i.e. managed by different data management systems such as RDBMS or XML DBMS, has long been studied in the context of multidatabase systems (also called federated database systems, or more recently data integration systems). Most of the work on multidatabase query processing has been done in the context of the mediator-wrapper architecture, using a declarative, SQL-like language. The mediator-wrapper architecture allows dealing with three major properties of the data sources: distribution (i.e. located at different sites), heterogeneity (i.e. with different data models and languages) and autonomy (i.e. under local control).

The state-of-the-art solutions for multidatabase query processing can be useful to transparently access multiple data stores in the cloud. However, operating in the cloud makes it quite different from accessing data sources on a wide-area network or the Internet. First, the kinds of queries are different. For instance, a web data integration query, e.g. from a price comparator, could access lots of similar web data sources, whereas a cloud query should be on a few but quite different cloud data stores and the user needs to have access rights to each data store. Second, both mediator and data source wrappers can only be installed at one or more servers that communicate with the data sources through the network. However, operating in a cloud, where data stores are typically distributed over the nodes of a computer cluster, provides more control over where the system components can be installed and thus, more opportunities to design an efficient architecture.

These differences have motivated the design of more specialized multistore systems (also called polystores) that provide integrated access to a number of cloud data stores through one or more query languages. Several multistore systems are being built, with different objectives, architectures and query processing approaches, which makes it hard to compare them. To ease comparison, the authors divide multistore systems based on the level of coupling with the underlying data stores, i.e. loosely-coupled, tightly-coupled and hybrid. *Loosely-coupled* systems are reminiscent of multidatabase systems in that they can deal with autonomous data stores, which can then be accessed through the multistore system common language as well as separately through their local language. *Tightly-coupled* systems trade autonomy for performance, typically in a shared-nothing cluster, so that data stores can only be accessed through the multistore system, directly through their local language. *Hybrid systems* tightly-couple some data stores, typically an RDBMS, and loosely-coupled some others, typically HDFS through a data processing framework like MapReduce or Spark.

A cloud architecture typically consists of multiple sites, i.e. data centers at different geographic locations, each one providing computing and storage resources as well as various services such as application (AaaS), infrastructure (IaaS), platform (PaaS), etc. To provide reliability and availability, there is always some form of data replication between sites. Although useful, the solutions devised for multidatabase systems (also called federated database systems) or Web data integration systems need to be extended in major ways to deal with the specific context of the cloud. This has motivated the design of multistore systems (also called polystores) that provide integrated or transparent access to a number of cloud data stores through one or more query languages. As NoSQL and related technologies such as Hadoop and Spark, multistore systems is a recent, important topic in data management, and we can expect much evolution in the coming years.

2.4.5 Comparison between Relational and Non-relational Databases

A **comparison** between relational databases and non-relational databases is covered in the paper by Vicknair [3]. It focuses on the underlying technology for the development of a data provenance system. It takes into account different objective and subjective measures for MySQL (relational) and Neo4j (NoSQL-Graph) databases. The objective tests include processing speed based on a predefined set of queries, disk space requirements, and scalability. Subjective tests include maturity/level of support, ease of programming, flexibility, and security. The conclusion of the paper is that using the graph storage for a production environment seemed premature due to lack of support back then (2010). Nevertheless, this might not be the case now, as these technologies broadly evolved in the later years and they gained more support and feedback from the user community.

Another comparative study of relational and non-relational database models in a web-based application analyzes MongoDB (non-relational) and MSSQL (relational) [4]. The operations that were undertaken are the standard ones: insert, update, delete and select. For inserting, updating and deleting records, MongoDB proved to perform better in terms of speed, especially when it comes to working with thousands of records at once. For selecting records, it is the other way around, so MSSQL proved to be faster. The conclusion of the study is that relational

databases are suitable for small scale applications, when they need to handle a limited amount of data.

The survey by Nishtha Jatana and al [7] affirms that the relational model is beneficial when it comes to reliability, flexibility, robustness, scalability requirements but in order to cater to the needs of modern applications where the data is huge and generally unstructured, non-relational databases show true signs of usability. The reliability of the database model is checked with the help of ACID properties. *Atomicity* stands for 'everything or nothing'. If any part of the transaction is left incomplete then the entire transaction is considered failed. *Consistency* ensures that a database before and after any transaction is stable at a valid state. *Isolation* ensures that multiple transactions executing at the same time do not affect one another's execution. Thus, requiring the concurrent transactions to be serialized. *Durability* ensures that once a transaction has been committed it will remain in the same state i.e. stored permanently even if there are some errors, or even if the system crash or power loss occurs.

The paper by Venkatraman [11] underlines the advantages for the NoSQL databases movement related to the Big Data development, compared to the SQL approach: NoSQL databases are a better option for the information systems that require high performance and dynamic scalability more than the requirements of reliability, highly distributed nature of the three-tier Internet architecture systems and cloud computing.

The Big Data advancements are also discussed in the paper by Acharya [12], mentioning the new era of NoSQL databases. The rapid growth in the digital world in the form of exponentiation to accommodate huge amounts of structured, semi-structured, unstructured and hybrid data received from different sources. By using the conventional data management tools, it is quite impossible to manage this semi-structured and unstructured data for which a non-relational database management system such as NoSQL and NewSQL are used to handle such types of data.

Presenting the advantages of non-relational databases, the paper of Vokorokos [13] indicates the ability to distribute them amongst servers and thus lowering the load on each individual server. It is better to divide the data amongst servers in the way that the single request will be processed only on one node.

In order to guarantee the integrity of data, most of the classical database systems are based on transactions [20]. This ensures consistency of data in all situations of data management. These transactional characteristics are also known as ACID (Atomicity, Consistency, Isolation and Durability). However, scaling out (i.e., scaling horizontally or adding more nodes to a system) of ACID-compliant systems has proven to be a problem. Conflicts are arising between the different aspects of high availability in distributed systems that are not fully solvable – known as the **CAP**-theorem. The CAP acronym stands for:

- Consistency: meaning if and how a system is in a consistent state after the execution of an operation. A distributed system is typically considered to be consistent if after an update operation of some writer all readers see his updates in some shared data source. There are nevertheless several alternatives towards this strict notion of consistency.
- Availability: and especially high availability, meaning that a system is designed and implemented in a way that allows it to continue operation (i.e., allowing read and write operations) if for instance nodes in a cluster crash or some hardware or software parts are down due to upgrades.

- Partition tolerance: understood as the ability of the system to continue operation in the presence of network partitions. These occur if two or more "islands" of network nodes arise, temporarily or permanently, which cannot connect to each other. Some also understand partition tolerance as the ability of a system to cope with the dynamic addition and removal of nodes (e.g. maintenance purposes).

The CAP theorem postulates that in a "shared-data system", only two of the three CAP characteristics can be achieved fully at the same time. For a growing number of applications and use-cases (including web applications, especially in large and ultra-large scale, and even in the e-commerce sector), availability and partition tolerance are more important than strict consistency. These applications have to be reliable which implicates availability and redundancy (consequently distribution among two or more nodes, which is necessary as many systems run on cheap, commoditized and unreliable machines and also provides scalability). These properties are difficult to achieve with ACID properties, therefore approaches like BASE are applied. Other properties relating to NoSQL technologies include, among others, sharding (i.e., Horizontal partitioning by some key and storing records on different servers in order to improve performance), Horizontal scalability (i.e., Distributing both data and load over many servers) and Vertical scaling (i.e., Use of multiple cores and/or CPUs by a DBMS).

The BASE approach according to Brewer [21] forfeits the ACID properties of consistency and isolation in favor of “availability, graceful degradation and performance”. The acronym **BASE** is composed of the following characteristics: Basically available; Soft-state; Eventually consistent. With regards to databases, Brewer [21] concludes that current databases are better at consistency than availability, and that wide-area databases can’t have both–a notion that is widely adopted in the NoSQL community and has influenced the design of non-relational data stores. Systems that can be characterized by the BASE properties include Amazon’s Dynamo, which is available and partition-tolerant but not strictly consistent, i.e writes of one client are not seen immediately after being committed to all readers. Google’s BigTable chooses neither ACID nor BASE but the third CAP-alternative being a consistent and available system and consequently not able to fully operate in the presence of network partitions. The CAP theorem was formulated to describe a major characteristic of distributed systems. In an era where scalability out of ACID properties was deemed inefficient, the CAP theorem was used as an argument in favor of adopting NoSQL databases. A rising number of database specialists have seeked to debunk this argument based on the fact that it is now possible for scalability to be maintained along with a strong consistency (e.g. NewSQL systems).

For NoSQL Databases, the argument holds still. The scalability provided by NoSQL comes at the ’small’ cost of having to prioritize availability over consistency in the existence of a network partition. Evidently, when the system is functioning in its normal state, the database can and should be able to provide both characteristics.

2.4.6 Query Languages

Generally speaking, a query language is a specialized programming language for searching and changing the contents of a database. Even though the term originally refers to a sublanguage for only searching (querying) the contents of a database, modern query languages

are general languages for interacting with the DBMS, including statements for defining and changing the database schema (if there is one), populating, searching or updating the contents of the database, defining integrity constraints over the database, defining stored procedures, defining authorization rules, defining triggers, etc.

Query languages have come a long way during the last few decades. The first database query language, SQL, was formally introduced in the early seventies following the earlier proposed and well-received relational model. SQL has influenced the design of dozens query languages, from several SQL dialects, to object oriented, graph, columnar, and the various NoSQL languages. These query languages are implemented and used in an unprecedented variety of storage and data management systems. In order to leverage the advantages of these solutions, companies and institutions are choosing to store their data in different representations, a phenomenon known as Polyglot Persistence. As a result, large data repositories with heterogeneous data sources are being generated (also known as Data Lakes), exposing various query interfaces to the user. Integrating this heterogeneous data (Big Data Variety) into a unified format and system, as has historically been the case with e.g., data warehouses, is nowadays becoming irrelevant. This is because (1) data is very large in size (Big Data Volume), (2) companies are less likely to sacrifice data freshness especially with the advances in streaming and IoT technologies (Big Data Velocity) [21]. While SQL, for example, comprises a sophisticated data structuring and very expressive query language, NoSQL trades schema and query expressivity for scalability. As a result, since no optimal representation exists, different storage and query paradigms have their right to exist based on the requirements of various use cases.

The paper by Mami [21] provides an extensive overview of the most popular querying languages in four database categories (relational, graph, hierarchical and document-oriented databases) and a review of the translation methods that exist between them. One of the conclusions is that, looking at the language scope coverage, there seems to still be a lack in covering the more sophisticated operations of query languages, e.g., more join types and temporal functions in SQL; blank nodes, grouping and binding in SPARQL. Such functions are motivated by and are at the core of many modern analytical and real-time applications. Indeed, some of those features are newly-introduced and some of the needs are only recently exposed, in which case the authors make the call to both update the existing works and build new solutions to embrace the new features and address the new needs. After discovering and exploring the various query translation methods, it appears that SQL and SPARQL are the most suitable languages to act as a ’universal’ language for realizing the heterogeneous data integration. SQL is the oldest query language with ever-continued development cycles and adoption. SPARQL is the stable query language of the so-called ontology-based data integration and access, which specializes specifically in integrating data coming from heterogeneous sources.

Squerall is a tool that allows the querying of heterogeneous, large-scale data sources by leveraging state-of-the-art Big Data processing engines: Spark and Presto. Queries are posed on-demand against a Data Lake, i.e., directly on the original data sources without requiring prior data transformation. The paper by Mami [22] showcases Squerall’s ability to query five different data sources, including inter alia the popular Cassandra and MongoDB. In particular, it demonstrates how it can jointly query heterogeneous data sources, and how interested developers can easily extend it to support additional data sources. During the last four decades, a variety of data storage and management techniques have been developed in both research and

industry. Today, we benefit from a multitude of storage solutions, varying in their data model (e.g. tabular, document, graph) or their ability to scale storage and querying. There are dozens of continuously evolving storage and data management solutions; for the NoSQL family Cassandra, MongoDB, Couchbase, HBase, Neo4j, DynamoDB, are just a few examples. However, those systems do not inter-operate, every stored data is locked in the respective system. For example, an ecommerce company might store Product information in a Cassandra database, Offers information in MongoDB, to benefit from its capability to store hierarchical multi-level values, and information about Producers obtained from an external source in a relational format. Without transforming and moving the data into a unified (scalable) data management solution, the data can hardly be explored and business insights be extracted using ad hoc uniform querying. The authors have taken on the mission of bridging this gap and developed Squerall, a software that allows to query heterogeneous data directly in its original form and source. They have used standardized and time-proven Semantic Web techniques to enable the uniform querying of heterogeneous data and support the mapping of terms in the original data to terms in higher-level ontologies, and the querying of the resulting uniform view using SPARQL.

Similar efforts to integrate and query large data sources exist in the literature. For instance, [24] defines a mapping language to express access links to NoSQL databases. [25] allows to run CRUD operations over NoSQL databases. [26] proposes a unifying programming model to directly access databases using get, put and delete primitives. [27] proposes a SQL-like language containing invocations to the native query interface of relational and NoSQL databases. [28] is a hybrid platform with consideration for both heterogeneous and dynamic data sources (streams). However, Squerall offers the highest number of supported data sources while providing the richest query capability, including joining, aggregation and ordering. The authors demonstrate Squerall's ability and efficiency in querying five different data sources (namely: CSV, Parquet, Cassandra, MongoDB and MySQL), and how it can be easily extended to support additional data sources, through several application scenarios.

Squerall addresses the Variety challenge of Big Data, which remains poorly addressed, by making use of Semantic Web standards and best practices. Squerall can conveniently be (pragmatically) extended to embrace new data sources, by making use of the query engines' own wrappers. This approach solves one of the most tedious tasks acknowledged across the literature, i.e., handcrafting wrappers. For example, in addition to the five sources evaluated in the paper, other sources like Couchbase or Elasticsearch can also be easily supported. As a result, Squerall is both innovative and unique in its capability to support a high number of data source types. Additionally, Squerall has been integrated into SANSA, a framework for scalable processing and analysis of large-scale RDF data, widening its scope to also access non-RDF data sources.

2.5. Results for the Second Part of the Literature Research

Data and information affect the decision-making process on the various activities of an organization, as data-intensive decision-making is being increasingly adopted by businesses, governments, and other agencies around the world, many leaders are aware of the importance of the data, while these organizations devise various ways to deal with the challenges such data brings, data processing, data quality and governance programs should not be underestimated

[30]. Therefore, an insight into how data governance is seen in the academic world will be presented in the following section.

Today's competitive business environment requires companies to collect, analyze, and interpret enormous quantities of data to enable better, more informed decision-making. Fortunately, there is an abundance of data available for most companies. However, without proper data quality and governance guidelines, these data might prove useless. Data quality is a necessity when extracting meaningful information. For, without it, a company may be left guessing at a direction or making skewed decisions based on flawed data. So, in order to achieve high quality, the data needs to be accurate, available, understandable, and relevant to the problem being solved. Ultimately, impactful, data-driven decisions are necessary for organizational survival. Unfortunately, the decision-making capabilities of a company are made significantly more complicated when there is a lack of data confidence. This lack of confidence adversely affects organizational performance. (IBM, 2008).

The data landscape of large, multinational corporations is often historically grown and characterized by isolated stand-alone solutions. This results in inconsistent data definitions and reports. There is a lack of understanding what data exists and where it comes from. Communication problems lead to increased development costs and long project cycles. In this context, often data consolidation (e.g. data warehouse) initiatives are launched to facilitate harmonization [31].

Recent big data environments, also called "data lakes" tend to be built in a less structured way than traditional ones like data warehouses (DWH). The strength of file- or document-based data stores like Hadoop or MongoDB is that data models do not need to be defined upfront (so-called "late binding" or "schema on read"), but that does not mean that requirements analysis and modeling should be neglected. In fact, the need of an information catalog to organize the data in a data lake has frequently been raised.

As Attard and Brennan mention [34], data is quite popularly considered to be the new oil since it has become a valuable commodity. This has resulted in many entities and businesses that hoard data with the aim of exploiting it. Yet, the `simple' exploitation of data results in entities who are not obtaining the highest benefits from the data, which as yet is not considered to be a fully-edged enterprise asset. Such data can exist in a duplicated, fragmented, and isolated form, and the sheer volume of available data further complicates the situation. Issues such as the latter highlight the need for value-based data governance, where the management of data assets is based on the quantification of the data value. This quantification will provide an opportunity for evidence-based approaches to data governance.

Organisations and companies are increasingly relying on their data to become more competitive, for example, by having greater knowledge of their customers, by taking more informed decisions, by finding new innovative uses for the data, by controlling risks and cutting costs, and also by innovating upon this data. Such use of data assets enables companies to not only better achieve their goals, but also to improve their financial performance [34].

2.5.1 Data Governance

In the study by Zhang, Ning, and Q. Yuan [30], it is mentioned that In the last century, still some organizations do not know what data they have, how critical that data is, the sources that exist for critical data, or the degree of redundancy of their data assets, however, data governance is an important topic for any organization that acknowledges the importance of their business data as a foundation to their success in recent years. It is an area of corporate management that looks at decision-making and authority for data-related matters.

At present, more and more organizations realize the importance of the data, they believe that data can improve performance, create value, enhance competitiveness as well as cut cost. So many surveys conducted by the institution or the conferences conveyed the idea.

Recently, nearly two-thirds of respondents in the 2012 Big Data study confirm that the use and the analysis of information (including Big Data) is creating a competitive advantage for their organizations (M. Schroeck, et al 2012). Another recent survey conducted by the (E. Pierce, et al 2012) showed International Association for Information and Data Quality (IAIDQ) also that more than 70% respondents considered data a strategic asset. According to the 2012 IBM CEO study, the ability of an organization to obtain value from data is strongly related with performance, where outperforming organizations are twice as good as underperformers at accessing and drawing insights from data.

Clearly, with the rapid growth of digitized data inside and outside the organization, and with the increase of possibilities to access this data, organizations have become aware of the need for right use of their data, there is a great emphasis on data and deriving its value through effective governance.

Since data governance and management efforts and investments are on the rise, it is becoming increasingly relevant to identify the economic value of data and the return on investment. Data value has been used as a basis for organisational decision making on quality, but also as a part of automated control systems for data lifecycles and file retention. Failing to value data will result in a number of consequences such as retaining information that has little to no value, reduction in data usage, and leaving data investments vulnerable to budget cuts. Hence, data value is an aspect that plays a very important role in data governance. The issue is that although gaining recognition as a valuable asset, data has as yet resisted quantitative measurement, and data value is as yet mostly limited to be a notional value [34].

Realistically, it is quite doubtful to have a one size fits all data governance solution, as any data governance efforts must fit the specific organisation in question and cater for the needs of the business. That being said, current approaches lack the link between data assets and organisational value. Such a strategy is essential in exploiting data assets to achieve competitive advantage that provides both short and long term value, therefore ensuring business success and sustainability [34].

Examples of data governance processes that can be optimised include:

- Data storage: for example more valuable data can be stored in more reliable, more secure storage, whilst less valuable data can be stored using cheaper options;
- Data access: more valuable data, for example sensitive data, can be restricted to be used by employees with a higher user grade;
- Data acquisition: Data can be acquired depending on whether its value for the enterprise is worth its cost;

- Data standards: Data standards can be defined based on the existing data specifications, with the aim of achieving the data specifications that are required by the context of use;
- Data maintenance: Data curation or maintenance processes can be prioritised according to data value. For example, either prioritising data assets that are very valuable to the business, and therefore will result in the highest impact, or otherwise prioritising data assets that will benefit most from being maintained.

2.5.1.1 Definitions of Data Governance

Data governance (DG) refers to the overall management of the availability, usability, integrity, and security of the data used in an organization. Since the initial emergence of data governance as an important and fundamental issue to organizations, the data governance community and researchers have published several definitions of the data governance. Although the definition of data governance is still evolving, current usage describes this discipline as being a facilitator for managers to take control over all aspects of their data resource.

Weber, K (Weber, K et al 2009) said that data governance specifies the framework for decision rights and accountabilities to encourage desirable behavior in the use of data, to promote desirable behavior, data governance develops and implements corporate-wide data policies, guidelines, and standards that are consistent with the organization's object, strategy, values, and culture. From the above, we can find data governance contains many aspects, which formed from a convergence of procedures, technologies, processes, policies, responsibilities and decision making rights for the use of data in organizations.

Data governance is an ongoing process of monitoring, evaluating, and assessing data, its users, and database activity to better understand and control data risk (Neera Bhansali 2013), in addition to compliance with legal requirements of information security, most of the data governance also propose the security requirements, related to how personally identifiable information is secured and protected, and establish the role of information or data security officer to guarantee the secure activities such as data access.

Data governance initiatives may be aimed at achieving a number of objectives including offering better visibility to internal and external customers and compliance with regulatory laws.

In the paper by Attard and Brennan [34], DAMA International defines data governance to be the "exercise of authority and control (planning, monitoring, and enforcement) over the management of data assets". Data governance is therefore the management of data architecture, data quality, data security, data operations, etc. Tasks include the setting, monitoring and enforcing of policies, standards, and procedures; the coordination, maintenance, and implementation of data architecture; the acquisition of data assets and the monitoring of their costs, quality and security, and the creation of data ownership rights. Data governance therefore enables the effective use of data assets.

2.5.1.2 The Framework of Data Governance

Firstly, almost all published frameworks recommend that data governance should undertake the maturity assessment, to establish the current state of data management and control, such as Gartner, IBM, MDM Institute, ARMA International, and CMMI Institute's Data Management Maturity Model (Neera Bhansali 2013). For example, Supplier Kalido (Data Governance Institute 2011) offers consultancy services and data management software,

encouraging customers to first do a self- assessment of their data governance maturity with online tool, the criteria that cover three potential aspects (organization, process and technology) of data governance implementation, based on the results of the assessment, an enterprise then fall into one of four data governance maturity stages: Application-Centric, Enterprise Repository-Centric, Policy-Centric, and Fully Governed.

Learned from the previous published IT governance framework by Weill & Ross (Weill & Ross 2004), Khatri & Brown (Khatri & Brown 2010) propose a data governance framework using five interrelated data decision domains: data principles, data quality, metadata, data access and data lifecycle, this framework become the important reference for the latter research. Both academic and practical sources provide data governance as a universal approach – one that fits all enterprises alike, however, Weber (Weber et al 2009) states that there should not one approach to data governance because of the distribution of accountabilities, so he proposes a flexible data governance model made up of several roles (executive sponsor, chief steward, business data steward, and technical data steward), decision areas (or tasks) and assignment of responsibilities, which stress data quality roles and their interaction with DQM (Data Quality Management) activities.

DeStefano [42] believes that in order to mitigate the cost and risk of poor data quality and information leakage while maximizing the opportunities that data can provide, it is imperative that organizations establish a data governance framework. The importance of data quality is well established. Data quality is a key prerequisite in many key business operations and objectives across many industries. Examples of such are: global supply chain management, customer relationship management, strategic decision making, business intelligence, and compliance with regulations.

2.5.1.3 Data Quality Management

In the above mentioned framework, data quality is an important element. Data quality management (DQM) focuses on the collection, organization, storage, processing, and presentation of high-quality data (Wende 2007). Data Quality Management specifies planning and implementation of techniques and processes to cleanse and purify existing data, and to ensure that whatever data is entered in the organizational information systems conforms to quality specifications (Abrar Haider 2012).

Data quality is seen as the most important aspect influencing usability of data for business processes and reporting (Friedman and Smith 2011). Data quality in turn largely determines the effectiveness of the business processes, and also influences the reporting quality (Eppler and Helfert 2004).

Data governance is not just about improving quality of data alone. It is about managing the asset of the organization so as to enable a continuous improvement-based learning progression (Abrar Haider 2012).

By understanding how data is used, and how long it must be retained, organizations can develop ways to make usage patterns fit to the optimal storage media, thereby minimizing the total cost of storing data over its life cycle (Khatri & Brown 2010). By placing data according to the life cycle, complying with business needs, data can be used more effectively distributed across multiple resources, thus leading to improved storage utilization and reduced storage acquisition costs.

Dai [40] underlines that data quality plays an important role in data governance. Data is a valuable asset for customers, companies, and governments. Data profiling technology can improve data quality. Data profiling tools discover the pattern of datasets, including data cleaning, data integration, and data analysis.

2.5.1.4 Metadata Architecture

Metadata describes what the data is about and provides a mechanism for a concise and consistent description of the representation of data, it is needed to be addressed in data governance, in order to assure the data is interpretable, standardizing metadata provides the ability to effectively use and track information, helping interpret the meaning or “semantics” of data. Different kinds of metadata play different roles in the discovery, retrieval, collection, management and analysis of data.

Nargesian [39] argues that, unlike data warehouses or DBMS, data lakes may not be accompanied with descriptive and complete data catalogs. Without explicit metadata information, a data lake can easily become a data swamp. Data catalogs are essential to on-demand discovery and integration in data lakes as well as raw data cleaning. In addition to extracting metadata from sources and enriching data with meaningful metadata (such as detailed data description and integrity constraints), metadata management systems need to support efficient storage of metadata (specifically when it becomes large) and query answering over metadata.

An example of a metadata management system is Google Dataset Search (GOODS) that extracts and collects metadata for datasets generated and used internally by Google. The collected metadata ranges from dataset-specific information such as owners, timestamps, and schema to relationships among multiple datasets such as their similarity and provenance. GOODS makes datasets accessible and searchable by exposing their collected metadata in dataset proles.

Metadata discovery provides the data abstraction that is crucial to data understanding and discovery, yet opportunities remain in better extracting and connecting knowledge from lakes and incorporating this knowledge into existing (general or domain-specific) knowledge bases.

Schema mapping permits the exchange of information between data sets using different schemas and recent work permits mapping discovery over incomplete (or inconsistent) schemas and examples. In sample-driven schema mapping, users describe the schema using a set of tuples. To give users flexibility in describing a schema, in multiresolution schema mapping, the user can describe schemas using a set of constraints of various resolutions, such as incomplete tuples, value ranges, and data types.

2.5.1.5 Methodologies

The paper by Priebe and Markus [31] discusses the Business Information Modeling (BIM) methodology that addresses two challenges – agile, business-driven, stepwise design and development of data intensive IT solutions as well as governing data within and beyond those solutions – by introducing a semantic business information model as a central point of reference.

Unlike BIM and Accurity Glossary, the existing available tools are rather limited in combining mapping specifications with business models (glossaries) and data requirements (specification lineage). Furthermore, they are agnostic to a particular modeling approach and

hence based on generic terms and categories rather than providing modeling guidance through functionality like attribute definitions, explicit inheritance or composite attributes.

A logical or physical data model alone is not suitable to enable harmonization and reuse in data-intensive environments. In order to address those goals adequately, the technical data models have to be accompanied by a semantic business information model, capturing the definitions and business rules plus the mappings to the different representations of the corresponding data artifacts. As an answer to that need, the authors have developed a methodology called Business Information Modeling (BIM). The business information model represents unambiguous, robust definitions of business concepts and the linking of data to these concepts. The model is an organization-wide and unique catalog of information pieces that represent the requirements of all relevant user communities.

Besides the elements of the business model Subject Areas, Entities, Attribute Definitions and Attributes it is core of the methodology to also maintain business requirements (usually more coarse grained and less formalized than attributes) and map them to elements of the business model. The BIM can be seen in Figure 2.3.

In this methodology, the business information model acts as the central anchor point for mappings to various data layers (e.g. sources in DWH projects). In order to facilitate a model-driven development approach, more detailed mapping information is needed than just tagging an attribute to a certain source data field. Joins or selection criteria may be needed. The authors use the initial assignment (“tagging”) of attributes to source systems or tables as an indication for the mapper, which detailed mappings he needs to add.

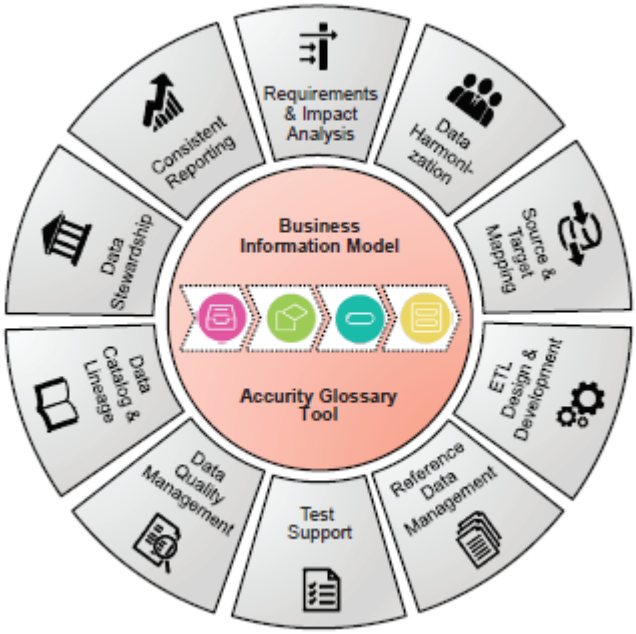


Figure 2.3 The Business Information Modeling (BIM) wheel proposed by Priebe and Markus [31]

If various layers in a data architecture are tagged with (or mapped to) elements of the business model, this information can be used to perform pragmatic data lineage queries. The

business information model is defined in a way that the attributes represent the most granular atomic pieces of relevant information. If the physical data representations are linked on this level of granularity, the lineage information is quite precise.

Now, how can a business information model be created? Unlike logical and physical data models, which can be acquired as pre-built industry reference models, a business information model is always organization specific, as the terminology and KPIs used will differ from organization to organization. Nevertheless, there are a number of sources that can help to define the organization-specific model. Depending on the setting, one source may be a reference data model of the respective industry (e.g. Teradata FSDM or IBM BDW). Further inputs that add content specific to the organization are corporate standards like a product catalog and existing legacy data models.

The authors base their business information model on the Entity-Relationship (ER) approach. As depicted by the four colored symbols in the center of the BIM wheel in figure 2.3, the main elements of their modeling approach are Subject Areas, Entities, Attribute Definitions and Attributes.

In organizations, where BIM has been applied within DWH project cycles, it has proven also very useful for governance. Data stewards and owners have been assigned on subject area, entity or even attribute level and stored as additional metadata. The tagging of data layers within the DWH environment is used not only during development, but also in production, e.g. to perform data lineage and impact analyses.

With the emergence of the big data hype, an old problem of IT, the integration of structured and unstructured information, is experiencing a renaissance. The “variety” aspect extends this to various data formats and stores, sometimes referred to as a “data lake”.

Ferguson [32] takes up on this and coins the term “data reservoir” arguing that data needs to be “refined” to be of business value. In particular, he argues that an “information catalog” is needed in order to:

- Document where data resides and came from;
- View data lineage;
- Name and describe data;
- Define shared business vocabulary terms;
- Classify data;

Picking up on this idea of an information catalog, the authors have developed the big data landscape scheme shown in Figure 2.4.

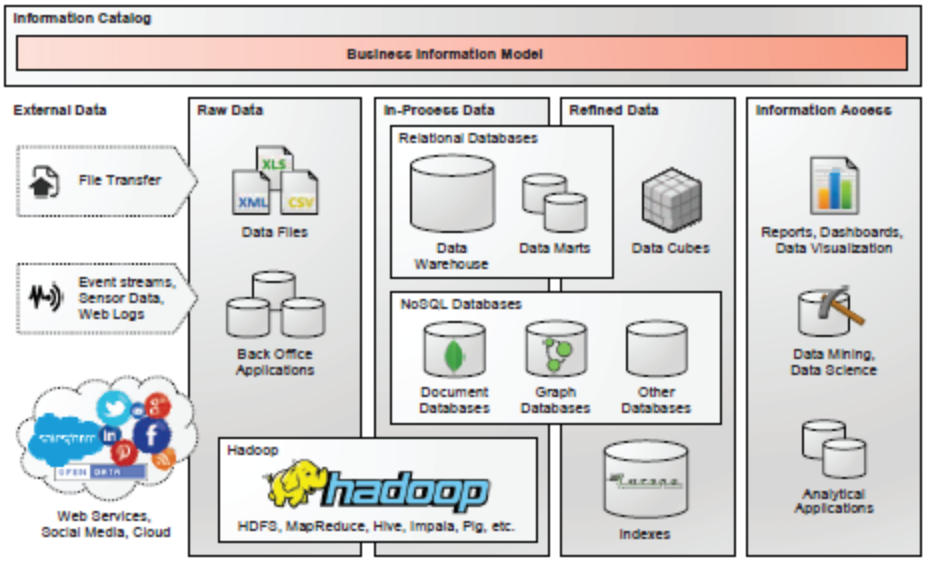


Figure 2.4 Big data landscape with BIM as information catalog

2.5.1.6 Use Case Examples

An implementation use case at an insurance company for a business intelligence competency center (BICC) as discussed by paper [32] shows that the underlying mission of the BICC was to empower strategic decision makers to accomplish important business objectives through a structure with strict data governance and information delivery. The BICC was to act as a liaison between analysts in the lines of business and the IT Data Management group. In order for it to work, the BICC needed to be business centric, rather than IT centric. This allowed it to better meet the needs of the lines of business it supports.

Prior to the creation of the BICC, no enterprise wide data governance policy existed. So, data analysts and business users experienced less than satisfactory results when analyzing data. Often, the development of reports and information processing were delayed by the lack of consistent, well-defined data. The BICC put forth a strong governance process to aid in the creation of impactful enterprise decisions. As such, the BICC sets-up and maintains definitions for data collected from the company's business insurance operations. It schedules and facilitates periodic meetings for its oversight committee, the Enterprise Data Governance Council.

A Data Governance Manager, who works in the BICC, was charged with chairing a data stewardship committee which is responsible for:

- Defining data across the enterprise;
- Improving data quality;
- Resolving data integration issues;
- Providing data usage policy and quality standards;
- Determining data security;
- Maintaining business rules applied to data and data retention criteria.

Now, the BICC provides support to data creation areas based on the knowledge gained through data quality issue resolution. It also prepares and publishes a data quality scorecard which enables the organization to gauge its progress on improving data quality. The BICC also works with the data stewards to help maintain data validation rules which enforce enterprise data standards, achieve key quality goals, and ensure root causes are addressed.

Finally, the BICC addressed analytical process sharing. Prior to the BICC, analysis performed throughout the organization was segregated by departmental units. By creating analytical silos, the company increased the amount of conflicting measures, redundant efforts, and decision making based on inaccurate information. The BICC catalogs the analytics dispersed throughout the company, validates the accuracy of the methods and results, and posts the analyses to an online repository which is available to all appropriate employees broadly.

Another use case example is covered by Attard and Brennan [34] for MyVolts, a company that uses data assets in order to obtain a competitive edge, as a use case with the aim of providing a first overview of challenges in value-driven data governance.

The first challenge is the quantification of the value of data as it is being acquired. MyVolts need to be able to measure the value of this data in order to identify whether this data is worth their effort and/or money. This quantification will not only enable MyVolts to reduce the risk of investing poorly in the data acquisition process, but also help target company efforts and aid decision making in the data exploitation and data curation processes. The first challenge is directly related to the second challenge; what makes data valuable? In this use case, in order to be valuable, data needs to be reliable, timely, relevant, accurate, with good potential for impact once its used, and preferably even unique (not available to other competitors). A further aspect of this challenge is the context of use; what might be valuable data for one use might be irrelevant for another. Therefore, different contexts of use will have different requirements as to what makes data valuable.

A successful effort to exploit data assets and achieve competitive advantage requires various data governance tasks, including the definition of roles; data policies, standards, and procedures; the definition of an interoperable data architecture; and data storage and organisation. Therefore, the authors here identify the need of implementing a value-driven data governance model. Such an evidence-based model would need to allow data assets to allow within the business or organisation, provide insight into what are the inputs and outputs of the existing processes, and also identify how these processes provide value to the business with regard to achieving the company goals.

2.5.1.7 Defining and Measuring Data Value

The paper by Attard and Brennan [34] discusses the data value concept. Data value is recognised as a “key issue in information systems management". Yet, while most research on information or data value seeks to identify dimensions that characterise it, there is still no consensus on the definition of data value. In fact, the multi-dimensional nature of value, as well as the role context plays in data value quantifying efforts, make the definition of data value quite challenging. The interdisciplinary nature of this field also adds to the complexity of this task.

Laney explores the applicability to the business and the availability to competitors as dimensions of data value [35]. Chen, on the other hand, presents an information valuation

approach that quantifies the value of a given piece of information based on its usage over time [36].

The measurement of data value in an unbiased manner allows for better data characterisation and classification, which then enable data exploitation optimisation. This requires monitoring data value dimensions within data value chains. Despite the growing literature on data as a valuable asset and on data exploitation, there is little to no work on how to directly assess or quantify the value of specific datasets held or used by an organisation within an information system. Moreover, existing methods for measuring the value of data often require intensive human e ort and are also case-specific.

Usage, cost, and quality are three recurring data value dimensions that are measured in existing literature. Chen, for example, devises an approach to measure data value based on usage-over-time [36]. This valuation method is derived from two measurable and observable metrics; usage and time. The author here infers the value of information based on a number of usage statistics that include usage count, usage time, the source of usage, and the purpose of the usage. Wijnhoven et al. extended Chen's usage-based data valuation approach with a utility-based estimation based on le metadata [37]. Through case studies in a consulting practice, they found that the frequency of use and the grade of user accessing the file were the most important predictors of value.

Various cost metrics are used in literature to measure data value. Stander breaks `cost' into two categories, namely: (i) the purchase price of the data asset, and (ii) the direct costs attributed to preparing the data for use [38]. Stander also mentions some approaches for measuring data value, including the cost approach, where data value is measured as the expenditure required to reproduce or obtain a data asset; the market approach, where data value is the price that organizations in the market are willing to pay for a data asset; and the income approach, which relies on the estimation of future income based on the exploitation of a data asset.

The existing literature therefore not only highlights the lack of existing metrics to quantify value, but also points out the need for more efforts in defining data value. Moreover, the literature also makes evident the complexity of quantifying data value, also due to its dependence on the context of use and its subjectivity. Yet, the subjective nature of some dimensions that characterize data value certainly does not rule out their quantification. Similar to some data quality aspects such as timeliness, such dimensions can still be accurately quantified in an objective manner, if only relevant for a specific context of use.

2.5.1.8 Data Profiling

Different authors give different definitions of data profiling.In the paper by Dai [40], it is described as “the set of activities and processes to determine the metadata about a given dataset.” or “data profiling is a process whereby one examines the data available in an existing database or flat file and collects statistics and information about that data. These statistics become the real or true metadata.” Data profiling can be utilized at different stages of data governance. Dai [40] believes that profiling is the process of verifying users’ structured data, semi-structured data, and unstructured data, gathering data structure, data pattern, statistical

information, distribution messages, and reviewing data attributes for data governance, data management, data migration, and data quality control.

Data profiling is included in multiple tasks for data governance. Scholars argue that people need to profile data only when cleaning, managing, analyzing, or integrating it; evaluating its quality; performing data archaeology; optimizing queries; or engaging in Extract, Transform and Load (ETL) projects. However, data profiling could also be used for compressing, verifying, masking, auditing, migrating, archiving, and recovering data as well as for generating testing data and data health reports.

Dai [40] argues that qualitative indicators are the best way to measure data quality and they present the following table:

Indicator	Definition
Accuracy	The degree to which data correctly describes the 'real world' object or event being described.
Completeness	The proportion of stored data against the potential of '100% complete'
Consistency	Similarity when comparing two or more representations of something against its definition.
Timeliness	The degree to which data represents reality at the required point in time.
Validity	Conformity of data's syntax (format, type, range) to its definition.
Uniqueness	Nothing will be recorded more than once based upon how it is identified.

Table 2.7 Indicators of data quality

DeStefano [42] argues that data is deemed high quality if it is fit for its intended use. While there are different definitions for "fitness", it is commonly implied to span the following 4 categories. It can be seen that some of the indicators presented in table 2.7 are also considered here:

- Accuracy: Determines the correctness of the data as it was recorded, against its actual value, based on the context of its usage.
- Timeliness: The recorded data must be current for its intended use.
- Completeness: The data values are persistent and it is adequate for depth and breadth. For example, this could mean having all the history for a given account, or all of the data values for a particular product.
- Credibility: Pertains to the trustworthiness of the data source.

In the paper by Dai [40], the authors come up with an example for a data profiling framework, that can be seen in table 2.8. It provides an overview of the layers that need to be considered, combined with the possible applications that could fulfill certain tasks.

Layer	Application
Web-UI Layer	User interface for ER model, business rules, KPI dashboard, batch or real-time job maintenance, input/output data source configuration.
Function Layer	Scores of data profiling missions (Section III).
Algorithm Layer	Utilize data mining, machine learning, and statistics, or other algorithms.
Parallelism Layer	Apache Spark for static data and full volume data; Apache Storm for real-time data.
Data Layer	Business rules, configuration data, metadata store in Hadoop, traditional databases.
Hardware Layer	Integrate CPU and GPU clusters for improving performance, especially real-time or machine learning tasks.

Table 2.8 An example of data profiling framework

2.5.2 The Accessibility of Data Lakes

The paper by Nargesian [39] gives a bit of context regarding data lakes and their accessibility. A data lake is a massive collection of datasets that: (1) may be hosted in different storage systems; (2) may vary in their formats; (3) may not be accompanied by any useful metadata or may use different formats to describe their metadata; and (4) may change autonomously over time. Enterprises have embraced data lakes for a variety of reasons. First, data lakes decouple data producers (for example, operational systems) from data consumers (such as, reporting and predictive analytics systems). This is important, especially when the operational systems are legacy mainframes which may not even be owned by the enterprise (as is common in many enterprises such as banking and finance). For data science, data lakes provide a convenient storage layer for experimental data, both the input and output of data analysis and learning tasks. The creation and use of data can be done autonomously without coordination with other programs or analysts. But the shared storage of a data lake coupled with a (typically distributed) computational framework, provides the rudimentary infrastructure required for sharing and reuse of massive datasets.

While some of the data in a lake is extracted, transformed, and loaded into existing database management systems (DBMS) or data warehouses, some of it may be exclusively consumed on-demand by programming environments to perform specific data analysis tasks. Moreover, the value of some of this data is transient, meaning additional analysis is required to create information with sufficient value to load into a data warehouse. Even though some of this data is not destined for traditional DBMS, there are still many open and fascinating data management research problems.

Current data lakes provide reliable storage for datasets together with computational frameworks (such as Hadoop or Apache Spark), along with a suite of tools for doing data governance (including identity management, authentication and access control), data discovery, extraction, cleaning, and integration. These tools help individual teams, data owners and consumers alike, to create and use data in a data lake using a self-serve model. Systems like IBM's LabBook proposes using the collective effort of data scientists to recommend new data visualization or analysis actions over new datasets or for new users. A grand challenge for data lake management systems is to support on-demand query answering meaning data discovery, extraction, cleaning, and integration done at query time over massive collections of datasets that may have unknown structure, content, and data quality. Only then would the data in data lakes become actionable.

An example of a data lake management architecture is shown in Figure 2.5 and is discussed in Nargesian [39]. The data sources may include legacy operational systems (operating in Cobol or other formats), information scraped from the Web and social media, or information from for profit data brokers. Operational systems often export all data as strings to avoid having to deal with type mismatches. The actual type information and metadata may be represented in numerous different formats. Other data may be pure documents, semi-structured logs, or social media information.

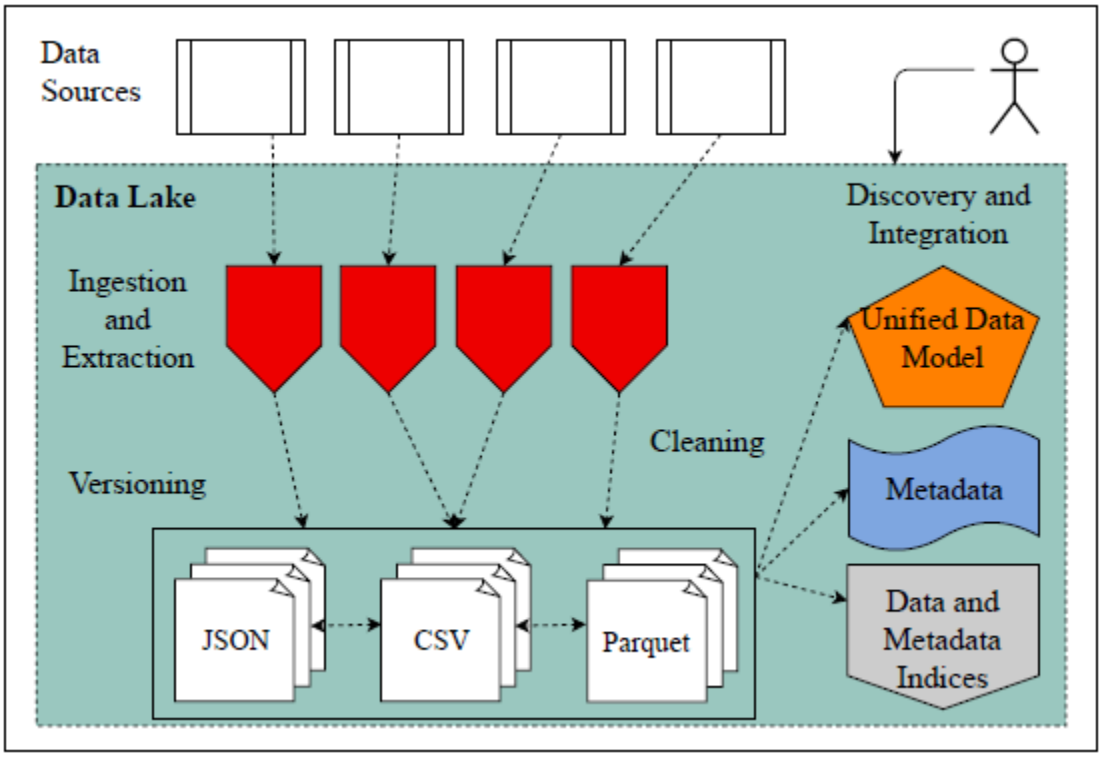


Figure 2.5 Example data lake management system

Kachaoui [41] believes that data lakes are merely means to an end. To achieve the end goal of delivering accurate and consistent business insights repeatedly, the aid of DW and data-driven management will be needed. DL's capability of storing and processing data at a low cost

has made it a perfect place for ETL; it is a data preparation process for business use. Data Lakes ingest all kinds of data by using a robust programming framework and low level coding language. All of these characteristics make DL a natural fit for ETL. It is considered as a new approach of analytical insights creation for businesses, from the acceleration of traditional enterprise reporting through new analytics driven by data science. It especially aims to bridge the gap between the rigidity of Data Warehouses/ data marts and the velocity and business needs.

This paper also presents an example of an Architecture for a Hybrid System that can be seen in Figure 2.6:

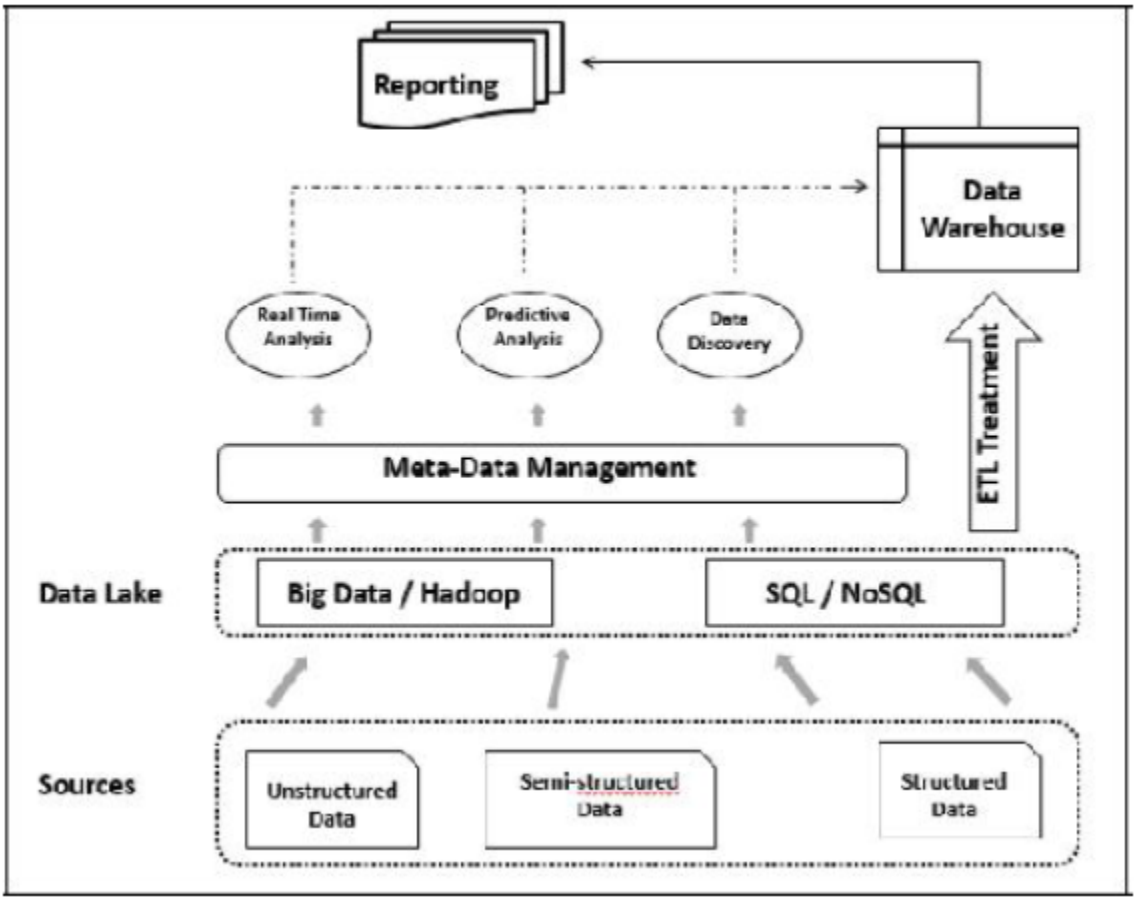


Figure 2.6 An example of architecture for a Hybrid System

2.5.2.1 Data Cleaning

Nargesian [39] mentions that while cleaning enterprise data has received significant attention over the years, little work has been done on cleaning within the context of a data lake. Logical and relational data cleaning typically requires correct schema information including integrity constraints. However, in data lakes the data may be stored in schema-less heterogeneous formats or using schemas that are only specified at application level. Although enriching data with schema information is one of the main goals of metadata management systems, postponing cleaning to the later stages of data processing may result in the propagation of errors through operations such as discovery and integration.

2.5.2.2 Dataset Discovery

Nargesian [39] discusses that, due to the sheer size of data in data lakes and the absence or incompleteness of a comprehensive schema or data catalog, data discovery has become an important problem in data lakes. To address the data discovery problem, some solutions focus on generating and enriching data catalogs as well as facilitating search on them. The authors consider these below with other data lake metadata management techniques. Other solutions operate on raw data (and existing metadata) to perform discovery. In query-driven discovery, a user starts a search with a query (dataset or keywords) and the goal is to find similar datasets to the query or datasets that can be integrated (joined or unioned) with the query.

3. Requirements

After completing the background research by conducting the literature study, enough expertise information was found in order to be able to start with the process of implementing a data management platform as a prototype, called “Data Observability prototype”. Before any development activities can take place, the requirements need to be determined, which are a description of features and functionalities of the target system to be developed. This step is part of the “Treatment design” and “Treatment validation” from the engineering cycle proposed by Wieringa [44], and it will answer the research question 2 mentioned in this Master thesis, *“What are the needs of the business users regarding this data management architecture?”.*

Firstly, an overview of the stakeholders and their goals should be known. There are two parties involved, the platform provider and the platform consumer. Within each category, there are more roles that can be identified with the help of the model provided by Alexander [45]:

- normal operators (also called “end users”, from the consumer side);
- functional beneficiaries (from the consumer side);
- maintenance operators (keeping the system running, from the provider side);
- operational support (from the provider side);
- interfacing system (from the provider side);
- developers (from the provider side);
- threat agent (a possible hacker);

In order to define the requirements for the “Data Observability prototype”, a survey was shared with a number of experts within the Dutch collaborative company. Their input is of high importance for being able to define a solution design that will reach the goal of the project.

The questions and full answers can be found in Appendix A. After analyzing the responses, the following section describes different personas (business users) applicable for the prototype, along with user stories suitable for each of them.

The entire requirements identification process is based on the Agile methodology. The Agile methodology for developing software evolved in the 90s as part of a reaction against traditional approaches that were considered heavy, bureaucratic and not adequately supported the activities of developers [48]. Agile is based on self-organization rather than rigid management practices, and the ability to manage to constant change rather than being blocked by rigid waterfall development process.

Agile processes always begin with the user or customer in mind. Nowadays, they are defined with user personas to illustrate different roles in a workflow the software is supporting or different types of customer needs and behaviors.

According to Williams, Clay, et al [49], the key stakeholders include not only those typically responsible for software development, but also stakeholders not typically involved in software engineering discussions. There are several stakeholder categories defined, and they can be seen in the figure below:

Stakeholder	Concerns
Customer	Obtaining an affordable product that provides the required features at an acceptable level of quality.
Developers	Understanding the requirements on the system they are building, as well as the technical, temporal, and business constraints on the development effort.
Brand Executives	Understanding the benefits, costs, and tradeoffs associated with the product in order to allocate resources optimally.
Legal	Ensuring that the delivered product satisfies any necessary regulatory and legal requirements, and that any third party code is appropriately managed.
Marketing	Creating collateral, shared assets, and demos to emphasize the value proposition and capabilities of the product being offered.
Product Manager	Coordinating all the stakeholders to deliver a product that meets all requirements within budget and the appropriate time frame.
Sales	Obtaining a balanced perspective on both technical usage and business applicability that supports productive conversations with customers.
Services	Integrating the software with other solutions and systems in customer enterprises.
Strategy	Mapping insights from the broader space of opportunities to the product to help determine future capabilities. Exploring to discover previously unknown opportunities.
Support	Ensuring that customers can properly utilize the required features of the software and integrate the software into a wide variety of existing systems.

Figure 3.1 Enterprise stakeholders and their concerns

3.1 Personas, Scenarios and Expectations

In our case, after analyzing the results from the survey, and considering the small scale of the prototype that is developed, three different personas can be identified that are part of the Customer and Developer category described in the figure 3.1. They come with specific goals that they want to achieve, and the artifact will be designed while keeping these goals in mind. To get a better understanding on the background of the personas, some possible scenarios, expectations and phases that they go through, these details were added as well in the below section:

- **Business user from a logistics company**: an end user being able to visualize the data on the IPaaS platform in the last phase of the cycle. They could be someone responsible for order management, or someone from customer support that needs to be able to check details of orders/customers;

  Scenario: Alex from the Customer Support department within the logistics company wants to be able to view orders that were delivered last week, and look for a specific customer that left an unsatisfactory review on the company because he did not receive his products ordered.

  Expectations: have a search input where he can look for the name of the customer and retrieve the relevant order data, including the address, the items ordered, and customer details.

  Phases

  Data storage: Alex knows that order data is stored for 1 year for this logistics company.

  Problem: A customer encountered a problem with his order that went missing and left a negative review for the company.

  Search: Alex can look up this specific customer to see the details of his delivery and see what went wrong with it.

  Solution: Alex finds the customer and his order and can get in contact with the customer to settle the disagreement, because the customer provided a wrong delivery address.

- **Architect**: keeps close contact with the business user to be up to date with their needs; accesses the IPaaS platform in order to decide what data is stored (message type), the type of metadata stored (timestamp, last changed, source system, target system) and for how long it is going to be stored from the available options (retention time can be 1-7 days/ 30 days/ 1 year/ permanent). This persona is configuring these settings based on the desires of the business user;

Scenario: Arthur from the Expert Services department from a Consultancy company has weekly meetings with their business client from a logistics company to be up to date with the changes that need to be done regarding their IPaaS platform that they are using.

Expectations: Arthur expects that the IPaas platform has a specific user interface where he can change the settings needed to store data for a specific retention time.

Phases

Data storage: Arthur finds out about the need of the logistics company to store their order data for 1 year. Along with it, extra metadata is wanted to be saved, like the timestamp, the time when data was last changed, source system that initiated to data transfer, the target system that is going to receive the data.

Changes on the platform: Arthur updates the settings on the IPaaS platform for the logistics company, to reflect their wishes regarding storing the data and metadata associated.

Confirmation: Arthur receives confirmation from the logistics company that their data was stored and can be accessed from the IPaaS platform.

- **Data Scientist**: technical person that can use the collected data for reporting purposes and further analysis, like data patterns, trends, graphs and different statistics;

  Scenario: Laura is hired as a data scientist to analyze the data collected, generate graphs and reports with statistics about the orders, regarding who is the most valuable client, what is the most sold product, etc.

  Expectations: Laura expects that data is stored in a database from where it can be easily accessed and extracted, so she can use different tools to generate the graphs and statistics.

  Phases

  Data storage: Laura accesses the database where data is stored and writes a script to extract the data.

  Data cleaning and processing: Laura processes the collected data and keeps only the relevant information that will help her generate the graphs and statistics.

  Generate statistics and graphs: Laura wants to find out who is the most valuable client based on who ordered products with the highest value. She extracts this knowledge by aggregating all the order amounts, grouped by customer.

3.2 User Stories

From the customer side, there are some Agile user stories that are taken into account before the implementation phase. These are mainly describing the functional requirements that need to be considered when developing the “Data Observability prototype”.

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

These user stories are formulated based on this format: *As <persona> I want to <requirement> so that <goal>.*

No	Title	User story
1	Store data	As a Business user, I want to be able to store the business data, so that I can access it later.
2	Query data	As a Business user, I want to have the possibility to query my data, so that I can find specific information easily.
3	Visualize data	As a Business user, I want to be able to visualize the data that was stored, so that I can view its details.
4	See relations and navigate	As a Business user, I want to be able to see what relations are between the stored data and navigate through them so that I can better understand the connections.
5	Select message type saved	As an Architect, I want to be able to choose what message type is stored so that only relevant data is saved.
6	Select metadata saved	As an Architect, I want to be able to choose to store specific metadata besides the business data (timestamp, last changed, source system, target system) so that more context is added to it.
7	Select retention time	As an Architect, I want to be able to select for how long the data is stored so that the appropriate retention time is used.
8	Specify permission	As an Architect, I want to be able to specify who can access the data so that only authorized persons can use the functionality.
9	Query and analyze data	As a Data scientist, I want to have the possibility to query my data, so that I can find specific information easily, and further analyze it.
10	Visualize and generate statistics	As a Data scientist, I want to be able to visualize the data that was stored, so that I can view its details and further generate graphs with statistics.
11	See relations, navigate, and determine data patterns and trends	As a Data scientist, I want to be able to see what relations are between the stored data and navigate through them so that I can better understand the connections and determine the data patterns and trends.

Table 3.1 User stories

The user stories from 9-11, related to the Data scientist role, focused more on analyzing the data, providing statistics and reporting are out of scope for this assignment, due to the limited amount of time provided, but it is kept as reference for future work.

3.3 Non-functional Requirements

Some of the non-functional requirements are extracted from the survey shown in Appendix A, distributed among experts from eMagiz. They are split into several categories:

- *Speed (latency) for the search performance*: the speed at which the results are delivered to users should take maximum 1 second; this is achieved by using ElasticSearch as storage and search engine;
- *Security*: due to the fact that the eMagiz platform is used as starting point to store the data that is flowing within, the security is achieved by having a login system in place by enforcing the correct data access privileges to their users;
- *Reliability,* which is the quality of being trustworthy or of performing consistently well;
- *Maintainability*: the source code is easily maintainable due to the fact that control versioning is used and software development practices were followed. There are comments throughout the code for a better understanding of the logic. There have been several “Knowledge sharing and transfer” sessions within the company, to ensure the functionalities of the prototype are understood by the development company so maintainability can be achieved;

A prototype will be developed, saving the data that flows through the Dutch IPaaS platform eMagiz as starting point. The prototype is demonstrating a fictional use case based on Orders-Customers-Addresses, specific for a logistics company. If the solution will arise interest from potential business clients, it will be fully integrated in the platform by the development team at a later point. In Chapter 5: Implementation, it is explained how the whole prototype functions.

4. Architecture Specification Model

To be able to answer the third Research Question proposed in this Master project, *"How to design a data management architecture based on ElasticSearch and GraphQL for retrieval and reporting of client data?",* an architecture specification model was created. This corresponds to the Treatment Design phase from the Design Cycle proposed by Wieringa [44].

In order to design the architecture specification model, the Archi modelling toolkit for creating ArchiMate models was used. The ArchiMate modelling language is an open and independent Enterprise Architecture standard that supports the description, analysis and visualization of architecture within and across business domains. ArchiMate is one of the open standards hosted by The Open Group and is fully aligned with TOGAF [46].

The TOGAF Standard, a standard of The Open Group, is a proven Enterprise Architecture methodology and framework used by the world's leading organizations to improve business efficiency. It is the most prominent and reliable Enterprise Architecture standard, ensuring consistent standards, methods, and communication among Enterprise Architecture professionals. Those professionals who are fluent in the TOGAF approach enjoy greater industry credibility, job effectiveness, and career opportunities. This approach helps practitioners avoid being locked into proprietary methods, utilize resources more efficiently and effectively, and realize a greater return on investment [47].

Below, the representation of the proposed system architecture is provided. It contains elements from the business, application and technology layers, to give a better overview of all the specific features involved. Each element is explained, along with the documentation that is provided by Archi. To be noted that the elements with blue font are newly added to achieve the purpose of this assignment and can be differentiated from the already existing elements of the Dutch provider IPaaS's platform, eMagiz.

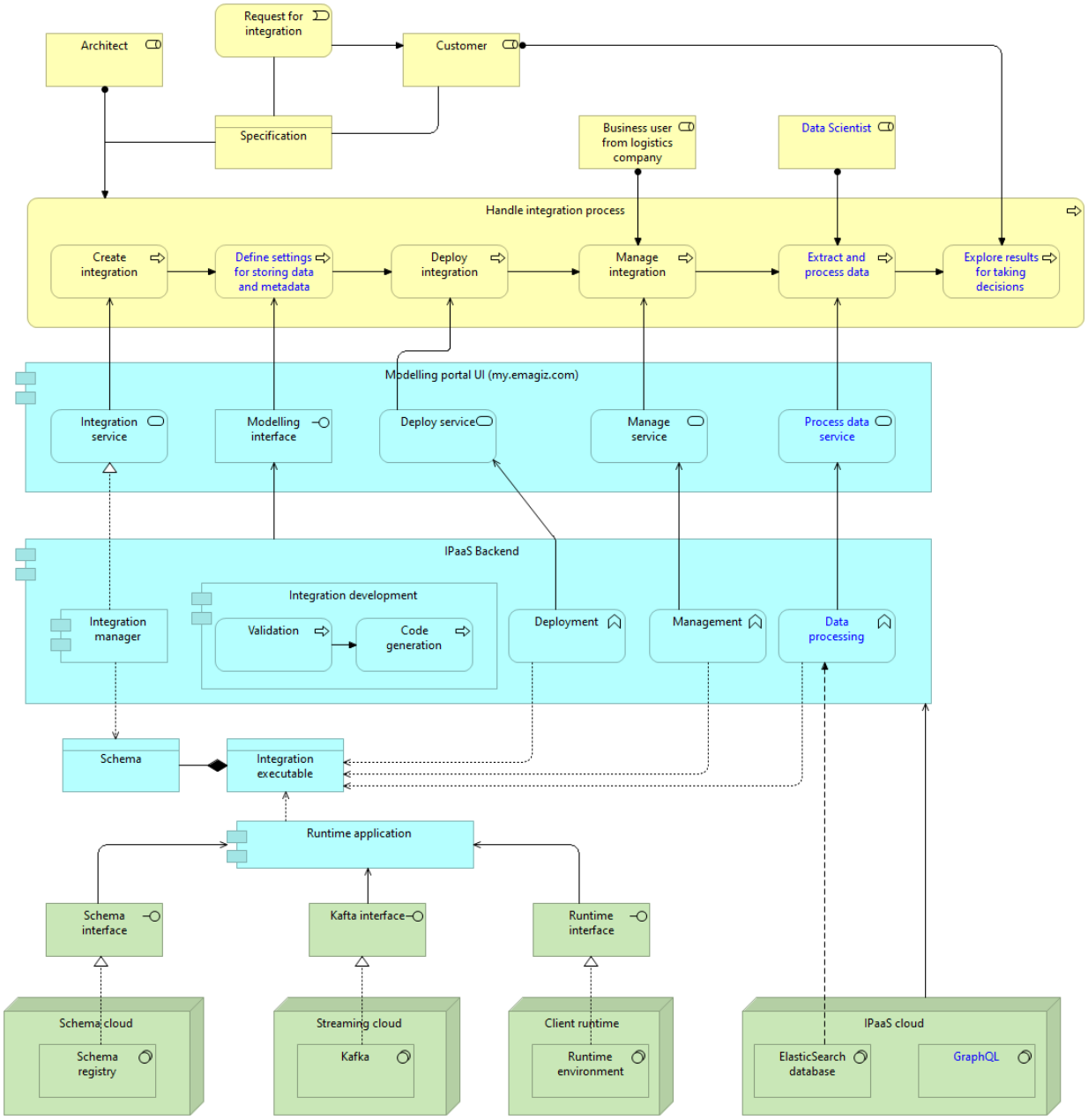


Figure 4.1 Architecture design by using Archimate

As a standard approach from Archimate, the business layer elements are colored in yellow, the application layer elements are turquoise and the technology layer elements are represented with green.

4.1 Business Layer

For the business elements represented with yellow, there can be 4 business roles that can be identified: *the Architect*, *the Customer*, *the Business user from logistics company* and the *Data scientist*. A Business Role represents the responsibility for performing specific behavior. A

Business Event represents an organizational state change and is most commonly used to model something that triggers behavior. In this model, the business event *Request for integration* is triggered by *The Customer,* who provides a *Specification* document to *the Architect*, on which the *Handle Integration* process will be based upon. This process contains 5 constituent processes, starting with *Create integration*, that triggers the *Define settings for storing data and metadata*, which triggers the *Deploy integration*, followed by *Manage integration*, *Extract and process data* and finally *Explore results for taking decisions*. A Business Process describes the internal behavior performed by a Business Role that is required to produce a set of products and services.

4.2 Application Layer

Going further into the application side represented with turquoise, the *Modelling portal UI* application component contains 4 application services: *the Integration service*, *the Deploy service*, *the Manage service* and the *Process data service*. There is also a *Modelling interface* that is part of this component. An Application Component represents an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable. An Application Component is a self-contained unit. As such, it is independently deployable, re-usable, and replaceable.

An Application Service exposes the functionality of components to their environment.
Another application component is the *IpaaS Backend* that contains the *Integration manager* component, *the Deployment*, *Management* and *Data processing* functions, and the *Integration development* component including *the Validation and Code generation* processes.
An Application Process represents a sequence of application behaviors that achieves a specific result and describes the internal behavior performed by an Application Component that is required to realize a set of services.

The *Schema* is an application object that results after the integration was created by the *Architect*, and it includes the *Integration executable* that is being accessed by the *Deployment*, *Management* and *Data processing* functions. A Data Object represents data structured for automated processing. A Data Object should be a self-contained piece of information with a clear meaning to the business, not just to the application level. *The Runtime application* is the last component from this section, and it accesses the Integration executable.

4.3 Technology Layer

Onto the technology layer depicted with green, there are three interfaces: *the Schema interface*, *the Kafka interface*, and *the Runtime interface*. A Technology Interface represents a point of access where technology services offered by a Node can be accessed. A Technology Interface specifies how the technology services of a Node can be accessed by other Nodes. A Technology Interface exposes a Technology Service to the environment.

There are 4 nodes that contain system software: *the Schema cloud* with *Schema registry*, the *Streaming cloud* with *Kafka*, *the Client runtime* with *the runtime environment* and the *IpaaS cloud* that has the *ElasticSearch database* and the *GraphQL* server. A Node represents a computational or physical resource that hosts, manipulates, or interacts with other computational or physical resources. Nodes are elements that perform technology behavior and execute, store,

and process technology objects. System Software represents software that provides or contributes to an environment for storing, executing, and using software or data deployed within it.

Figure 4.2 below illustrates the architecture proposed from a technology provider point of view. The Customer Bus is the integration pattern chosen for the prototype, which is explained further in chapter 5. The ElasticSearch database can be hosted on the cloud, while the GraphQL server and the IPaas gateway are backed by the AWS service. The personas defined (Architect, Business user and Data scientist) interact with the portal emagiz.com.

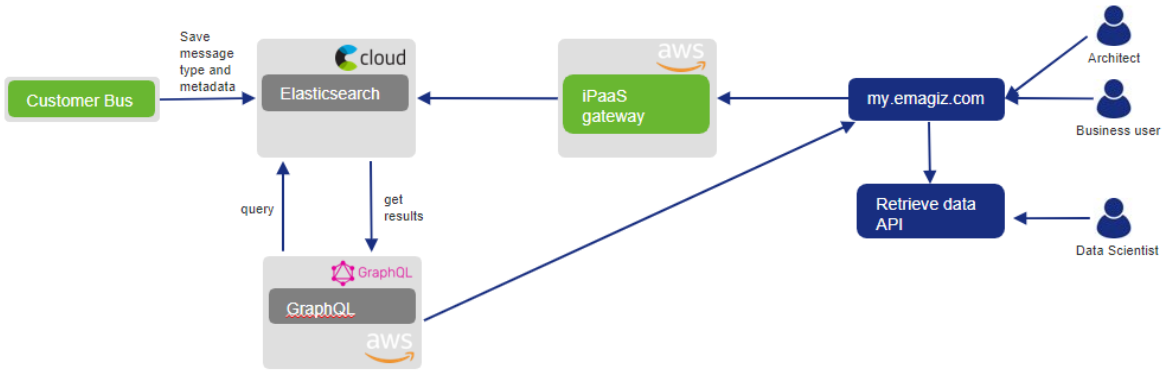


Figure 4.2 Technology provider architecture proposed

5. Implementation

As an answer to the fourth Research Question proposed by this Master project, *"How to integrate the data management architecture in the business context of a Dutch organisation?",* the Implementation chapter discusses broadly how the "Data Observability prototype" was developed and how it can be integrated into the eMagiz environment. This step corresponds to the "Treatment implementation" step from the Design Cycle proposed by Wieringa [44].

As it can be seen in the architecture model from the previous chapter, the existing integration platform from eMagiz can be enhanced with new features proposed in this Master project. These features mainly address the collection of data transported between various systems within the eMagiz platform on ElasticSearch, that acts as a database storage. Further on, GraphQL provides the schema visualization and navigation properties, that make understanding the data structure easier to accomplish. These enhancements can have various advantages for the final users: being able to visualize the data that is flowing within the platform for debugging and tracing purposes, having a search function to look through the set of data and find something specific, archiving or just utilizing the solution as a data management platform.

The prototype developed acts as a proof of concept for the technology merge of ElasticSearch and GraphQL and is not yet fully integrated into the eMagiz platform. The main reason for this decision is that during the timespan of the assignment at the company, a clear business client could not be found to adopt the solution stack. One of the causes for this stands behind the security and privacy concerns. Up to now, eMagiz acts as an integration platform that enables the transportation of messages between applications and systems, and the storage functionality falls under the responsibility of the source and destination parties. If this responsibility is taken over by eMagiz, as a database storage provider, then new regulations are in place that need to be considered.

The contents of this chapter will further explain how the artifact was implemented and what are its functionalities. The prototype is based on an Orders-Customers-Addresses business case, that resembles the application for a logistics company. The visualization part of the prototype is completed with React, which is a front-end JavaScript library for building user interfaces and UI components, represents a standalone solution. One of the advantages of this approach is that it can be integrated into eMagiz by the development team at a later point, as the web application is based on JavaScript.

5.1 eMagiz as IPaaS

Some background information about the eMagiz platform is needed for a better understanding of the solution design. As an Integration Platform as a Service, eMagiz is mainly used for integrating systems and applications with each other and playing the role of a broker between systems that produce and consume data of each other. eMagiz offers three main integration types with many supported integration patterns, that can solve any use-case as one firm technical fundament. These integration patterns are Messaging, API gateway and Event Streaming. Every integration pattern has its own scope, therefore supporting different use cases.

eMagiz offers the flexibility to deploy these patterns from just one platform. However, the power of an integration platform does not only lie in the technical component, but it also lies in the way in which the functionality is offered to the user. eMagiz offers its users rich management functionality that is aimed on documenting data, simplifying integration development and management by using a low-code visual platform, maintaining architecture and cloud environments and managing data. These features enable users to easily integrate robust applications that deliver value to the organization over a long period of time [50].

eMagiz approaches integrations from a lifecycle perspective, which consists of several phases, each with its own focus:

- **Capture**: Visualize the integration landscape and collect and document integration information;
- **Design**: Create a solution design, including a choice out of different integration patterns. Develop a Canonical Data Model and deployment architecture;
- **Create**: Realization of the solution design & modelling of integrations;
- **Deploy**: Establish the test, acceptance and production environment & deploy, test and accept integrations;
- **Manage**: monitoring of environments, transactions, performance, central error handling and notifications to administrators;
- **Improve**: Conduct trend analysis based on statistics and adjust environmental configurations;

Back to the implementation of the artifact, called “Data Observability prototype”, there are several areas that need to be completed for having a functional solution:

- Data storage in ElasticSearch from eMagiz;
- Schema visualization and navigation with GraphQL;
- Entity insight with GraphQL;
- Discovering related products with GraphQL;
- Data querying and results browsing;
- Integration of ElasticSearch and GraphQL into one web application: React;

5.2 Data Storage in ElasticSearch from eMagiz

The first step in the process revolves around storing the data. For this purpose, the eMagiz platform was used to create a local message bus that sends order data from System A to System B, that will automatically get saved in the ElasticSearch database, along with metadata. The metadata consists of the timestamp when the message was sent, the last changed timestamp of the message, the source and destination systems. Section 2.4.3 from this document extensively explains what ElasticSearch is and how it can be configured.

Using the local message bus is the preferred approach for implementing the “Data Observability prototype” because this is one of the most popular choices and the pick-up file function is well suited for the case. Nevertheless, the prototype could be used on multiple use cases, where data is intercepted and synchronized to the data storage.

The messaging pattern is used to exchange messages and data packages reliable, (as)synchronous and flexible between applications. With the help of a messaging integration, a message (= a data package to be sent between two applications) can be transformed and enriched in terms of content, format and protocol. This allows organizations to effectively integrate any application, regardless of the underlying technology.

While keeping in mind the CREATE phase from the integration lifecycle of eMagiz, the Archimate model from figure 5.1 depicts the process of storing messages and metadata in ElasticSearch while using the IPaaS platform. GraphQL is stacked on top of the database layer, and is used to provide the schema visualization and querying possibilities. React, which is a front-end JavaScript library for building user interfaces and UI components, is used to make the solution available to the user, who can interact with it by doing queries and visualizing the data.

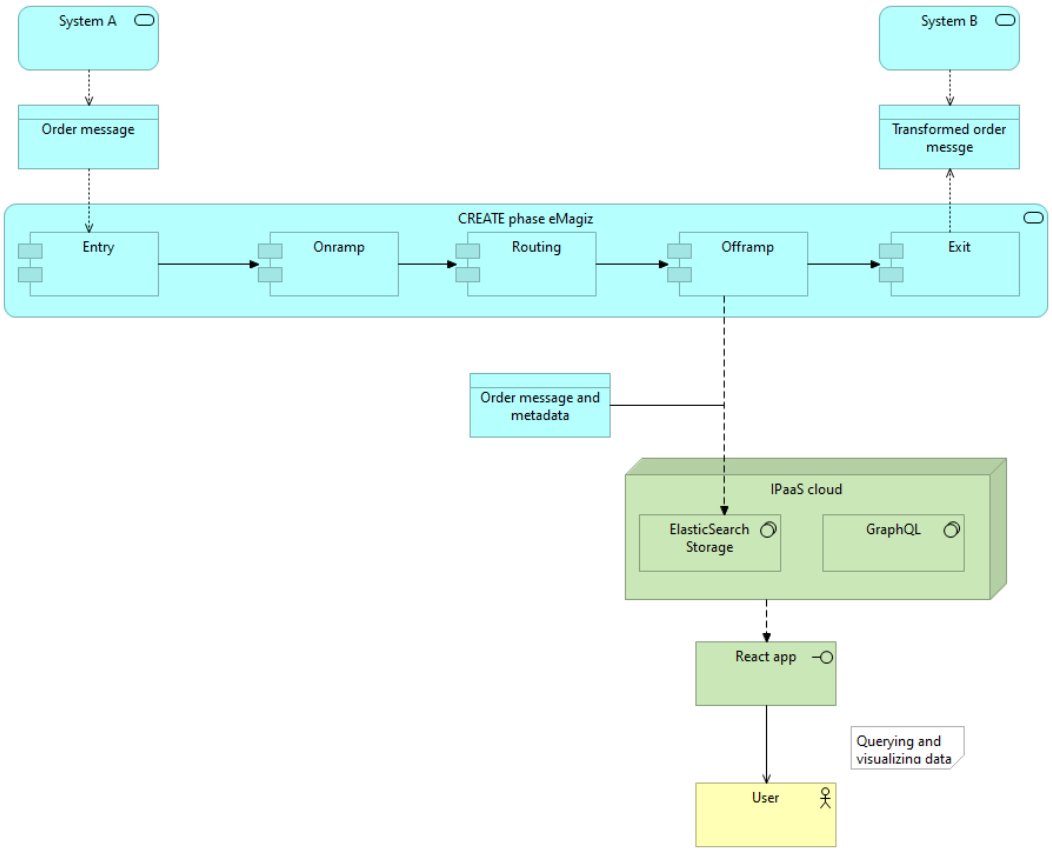


Figure 5.1 Overview of solution architecture

The application components of *Entry, Onramp, Routing, Offramp* and *Exit* are part of the CREATE step from the eMagiz platform. The *User* represents a generic actor that desires to query the data and visualize the data after it was saved in ElasticSearch.

The ElasticSearch endpoint is running on localhost, hosted in a Docker container and it saves records as JSON documents. An “emagiz_orders” index was created with the mapping structure shown in Figure 5.2, because the data structure of the message was previously known. Nevertheless, this is not a requirement for ElasticSearch, as the mapping is automatically created

when a new document is added to the storage. The name of the index needs to be defined because this is how the ElasticSearch endpoint is accessed by the eMagiz platform in order to save the data.

```
"Order" : {
    "Customer" : {
        "Email" : {"type" : "text"},
        "Name" : {"type" : "text"}
      },
    "Date" : {"type" : "date"},
    "DeliveryAddress" : {
        "City" : {"type" : "text"},
        "Country" : {"type" : "text"},
        "Name" : {"type" : "text"},
        "PostalCode" : {"type" : "text"},
        "Street" : {"type" : "text"},
        "StreetNumber" : {"type" : "text"},
        "Type" : {"type" : "text"}
      },
    "PickupAddress" : {
        "City" : {"type" : "text"},
        "Country" : {"type" : "text"},
        "Name" : {"type" : "text"},
        "PostalCode" : {"type" : "text"},
        "Street" : {"type" : "text"},
        "StreetNumber" : {"type" : "text"},
        "Type" : {"type" : "text"}
      },
    "OrderId" : {"type" : "text"},
    "OrderLine" : {
        "Description" : {"type" : "text"},
        "PackageUnit" : {"type" : "text"},
        "Quantity" : {"type" : "text"},
        "Weight" : {"type" : "text"}
      },
"creationDate" : {"type" : "date"},
"destination" : {"type" : "text"},
"lastChanged" : {"type" : "date"},
"sender" : {"type" : "text"}
}
```

Figure 5.2 ElasticSearch mapping structure

The official definition for an ElasticSearch index is: “An index is like a ‘database’ in a relational database. It has a mapping which defines multiple types. An index is a logical

namespace which maps to one or more primary shards and can have zero or more replica shards."

The last fields from the mapping (creationDate, lastChanged, destination and sender) are part of the metadata that gets added along the original message transported from System A to System B. This makes the sorting of data easier, based on the creationDate, which is a timestamp of when the message was transported through eMagiz.

To make a comparison between a relational database and ElasticSearch, the structure behind them can be represented like below, where DBMS is Database Management System:

Relational DBMS => Databases => Tables => Columns/Rows
Elasticsearch => Indices => Types => Documents with Properties

As part of the research, the wireframe design below was created with the intent to demonstrate how this new feature could be embedded in the design phase of eMagiz, where users are making design decisions on their data and integration.

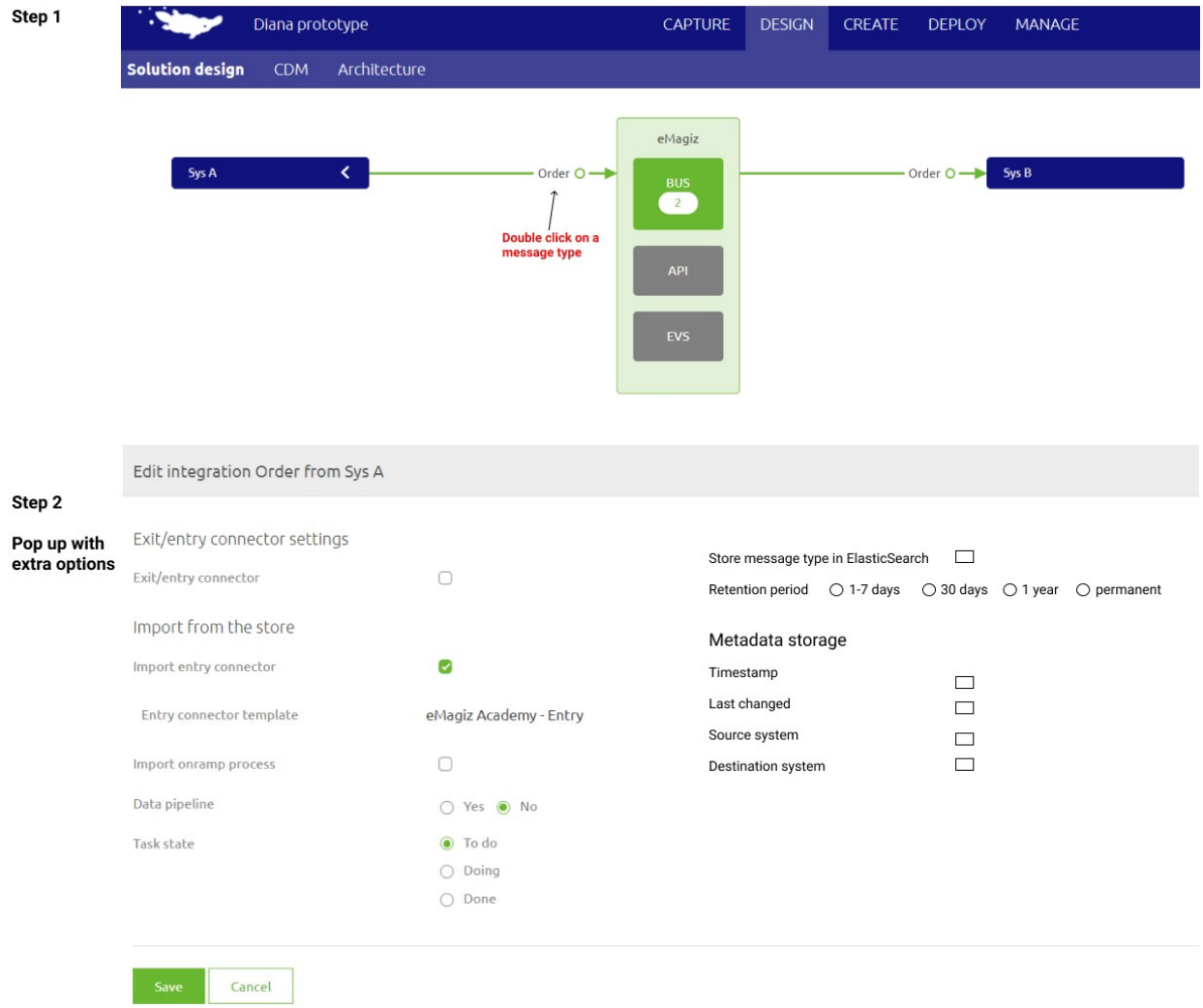


Figure 5.3 Wireframe design for saving a specific message type in ElasticSearch within eMagiz

This configuration of what messages get stored, what is the retention period and what kind of metadata gets generated is part of the Architect role described in Chapter 3: Requirements. All these features can be seen on the right side of the wireframe and are part of the proposition on how to integrate the functionalities into the existing eMagiz platform. This answers the 4th Research question, *"How to integrate the data management architecture in the business context of a Dutch organisation?"*

5.3 Schema Visualization and Navigation with GraphQL

A GraphQL server was set up by using the Express back-end web application framework for Node.js. It uses the same schema mapping as the one defined previously in the ElasticSearch index, to make sure that data can be successfully retrieved from the database layer. Then the connection to the ElasticSearch was added. This schema mapping is also known in eMagiz, as the platform holds the feature to specify domain models of the customer data landscapes.

One extremely useful tool that is open source and was incorporated in the React solution prototype is GraphQL-voyager [51]. It provides interactive visualization of the graph data model and exploration of the API and is represented in figure 5.4.

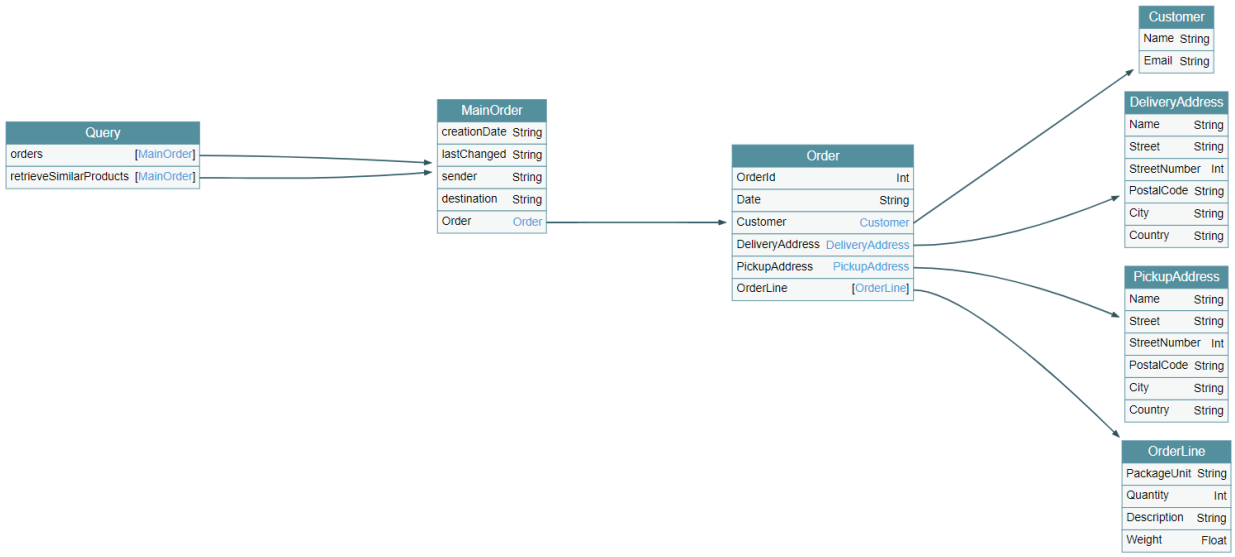


Figure 5.4 Schema visualization with GraphQL-voyager

This powerful tool provides an insight into how the data is structured, what entities exist and how they are related, what fields they contain and their data type. From the schema above, we can identify the Order entity as the root, and Customer, DeliveryAddress, PickupAddress and OrderLine as children entities.

The tool provides an interactive functionality, in the sense that if one entity is clicked, all the related entities will be highlighted. This tool has been further modified and enhanced to provide better insight into the data and the functionalities are explained in the further sections.

5.4 Entity Insight with GraphQL

One extension to the functionality of visualizing the entities from the schema mapping is being able to click on a specific one and explore the data stored in the fields belonging to it. This provides a table with all the results from that specific entity and the corresponding fields, as it can be seen in Figure 5.5.

GraphQL results after clicking on an entity from the schema visualization

Order

Customer Name	Order Date	Order Id	Customer Email	Delivery Address Name	Order Line Package Unit	Metadata Creation Date	Metadata Last Changed	Metadata Sender	Metadata Destination
Nancy Rose	05/09/2020	5	n@rose.com	Rose street	BOX	25/05/2021, 12:50:48	25/05/2021, 12:50:48	sysa	sysb
Robert Aang	05/07/2020	3	r@aang.com	Riverstreet	BOX	25/05/2021, 12:42:03	25/05/2021, 12:42:03	sysa	sysb
Diana Laura	05/04/2021	1	d@laura.com	Transportcentrum	BOX	25/05/2021, 12:38:03	25/05/2021, 12:38:03	sysa	sysb
Amelia Earhart	05/08/2020	4	a@earhart.com	Planestreet	BOX	25/05/2021, 12:49:08	25/05/2021, 12:49:08	sysa	sysb
Hannes Gerbert	05/09/2019	6	h@gerbert.com	Hengelosestraat	BOX	25/05/2021, 12:54:33	25/05/2021, 12:54:33	sysa	sysb
Marcel Proust	05/04/2020	2	m@proust.com	Main Street	BOX	25/05/2021, 12:40:38	25/05/2021, 12:40:38	sysa	sysb
Anne Marie	05/09/2018	7	a@marie.com	Pathmossingel	BOX	25/05/2021, 12:58:08	25/05/2021, 12:58:08	sysa	sysb
Paul Roberts	05/09/2017	8	p@roberts.com	Ripperdastraat	BOX	25/05/2021, 13:00:03	25/05/2021, 13:00:03	sysa	sysb
Norman Jackson	05/09/2019	9	n@jackson.com	Beukstraat	BOX	25/05/2021, 13:02:28	25/05/2021, 13:02:28	sysa	sysb
Joanna Lent	05/09/2018	10	j@lent.com	Florestraat	BOX	25/05/2021, 13:04:33	25/05/2021, 13:04:33	sysa	sysb
Bert Gogh	05/09/2020	11	b@gogh.com	Sumatstraat	BOX	25/05/2021, 13:06:58	25/05/2021, 13:06:58	sysa	sysb
John Richards	05/09/2021	12	j@richards.com	Nieuwestraat	BOX	27/05/2021, 09:54:50	27/05/2021, 09:54:50	sysa	sysb
Hans Reeds	05/09/2021	13	j@richards.com	Transportcentrum	BOX	28/05/2021, 16:42:15	28/05/2021, 16:42:15	sysa	sysb
Mary Jane	05/10/2021	14	m@jane.com	Highstreet	BOX	04/06/2021, 10:03:28	04/06/2021, 10:03:28	sysa	sysb

Figure 5.5 Entity insight for the Order entity, with all the corresponding fields

By being able to see the data structure from the visualization and retrieving the data stored in the entities is a strong combination for providing a good reporting solution, without the user needing to execute database queries.

5.5 Discovering Related Products with GraphQL

One of the strengths of GraphQL resides in the fact that the business domain is structured as a graph, which makes traversing the data straightforward. In the figure below, the data schema that is used for the prototype can be seen. It is based on Order as a root node, having the Customer, PickupAddress, DeliveryAddress and OrderLine as children:

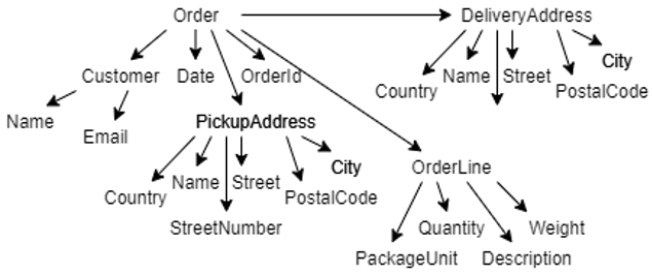


Figure 5.6 Data schema as a graph

To demonstrate the capability of retrieving related data, a GraphQL query is triggered to retrieve the OrderIds and Descriptions of related products when clicking on a product description field from the OrderLine entity. An example can be seen in the figure below, belonging to the scenario when a user clicks on the product description of a laptop, and all the related items are retrieved and presented in a popup:

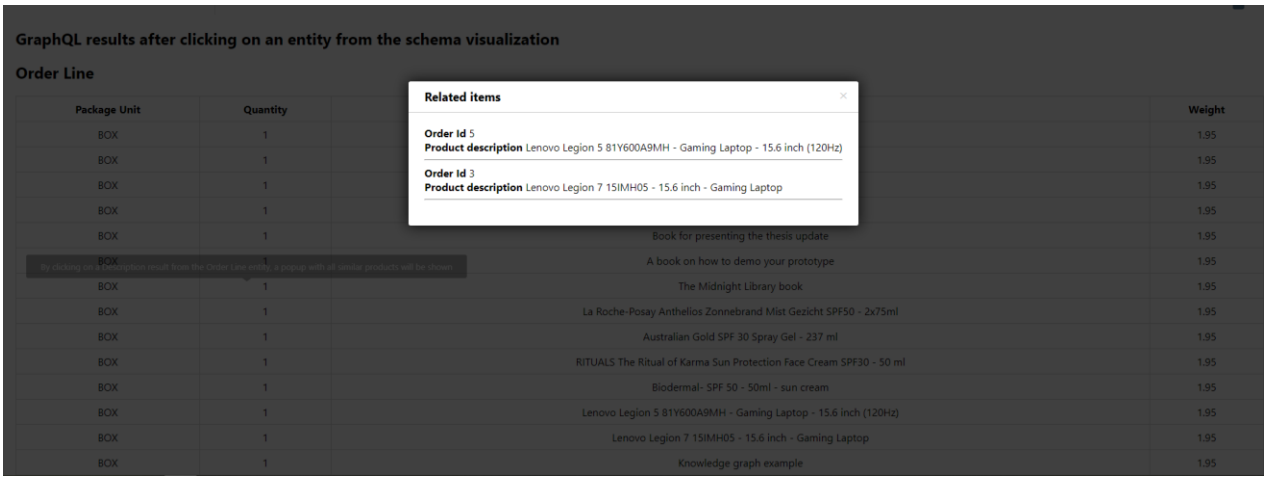


Figure 5.7 Results from the related products query

Being able to discover similar products at the click of a button provides powerful potential for a data management platform. The similar products are retrieved based on the text of the description. If there is a match with other orders, then they will get retrieved by the query.

The GraphQL query used to fetch the data saved in ElasticSearch is the following:

```
retrieveSimilarProducts(productDescription:"${productDescription}"){
  Order{
    OrderId
    OrderLine{
      Description
    }
  }
}
```

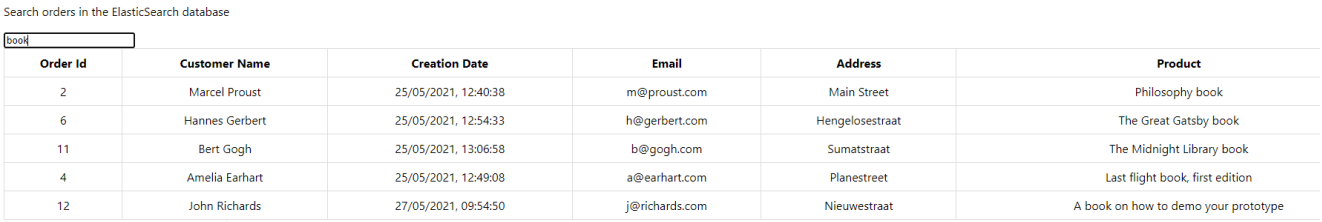
Figure 5.8 GraphQL query to retrieve similar products

The above query makes use of the “productDescription” variable that contains the text of the specific product description clicked by the user on the front end. The only fields retrieved are OrderId and Description. This request demonstrates the straightforward way of running a query with GraphQL, where only the required fields are retrieved.

5.6 Data Querying and Results Browsing

When it comes to data querying and results browsing, ElasticSearch represents the database where the Orders information is stored. React was used to build a user interface that provides a text search input, based on which a query is executed on the backend, and results are shown back to the user. The fields that are shown are considered the most relevant from all the entities, to be able to have a complete overview over the data. React was an advantageous choice due to the fact that other widgets in eMagiz are also based on React, and this will make the integration of the prototype into the platform easier.

All these orders have been transported within the integration platform, eMagiz, while using the messaging bus type of integration. What all the examples from figure 5.9 have in common is the “book” term that is part of the description of the product and was used to do the search. The search is performed on all the fields from all the entities saved.


Search orders in the ElasticSearch database

book

Order Id	Customer Name	Creation Date	Email	Address	Product
2	Marcel Proust	25/05/2021, 12:40:38	m@proust.com	Main Street	Philosophy book
6	Hannes Gerbert	25/05/2021, 12:54:33	h@gerbert.com	Hengelosestraat	The Great Gatsby book
11	Bert Gogh	25/05/2021, 13:06:58	b@gogh.com	Sumatstraat	The Midnight Library book
4	Amelia Earhart	25/05/2021, 12:49:08	a@earhart.com	Planestreet	Last flight book, first edition
12	John Richards	27/05/2021, 09:54:50	j@richards.com	Nieuwestraat	A book on how to demo your prototype

Figure 5.9 Query the term “book” on the ElasticSearch database

The query that is done on the backend server is a “match” query. There are several types of queries that operate in ElasticSearch. The overview [52] from the figure below provides an explanation on what functions and what kind of text would be selected according to each specific query name, along with examples for each case.

Query Name	Functions	Sample Query	Matching Text	Not Matching Text
match	• Simple match query. • Matches if one term is a match. • More the terms matched, the better the score. • Analyzer applied by default.	green elephant	• "green is what elephants want" • "elephants are great" • "green color are great"	• "white potato" • "yellow leaves"
phrase match	• Matches only if the terms come in the same order, in a consecutive manner.	"green elephant"	• "green elephant is here" • "white and green elephants"	• "elephant green" • "green is my elephant name"
prefix	• Similar to phrase query, it will search for the exact term and stores the document IDs.	"NY M"	• NY M545 • NY M-34	• NYM545 • NYM-34
term	• Queries on the tokens generated. • Analyzer is not applied.	"baskets"	• there are baskets in here	• there are Baskets in here • there are basketsin here
multi_match	• Applies match query on multiple fields.	field1: "roses" field2: "red"	If field1 contains text with the word "roses" or if field2 contains text with the word "red".	If both the fields do not contain any text matching both words.
bool	• Applies multiple queries .	must : "field1": "roses", must_not :"field2": "red"	Will match the documents with the word "roses" on field 1, but eliminate the documents with the word "red" on field 2.	If the given fields in the must field don't contain any matching text, or if the documents containing the words mentioned in must not field.

Figure 5.10 ElasticSearch Queries overview

Depending on the type of results that need to be retrieved and how strict the search should be, these different querying possibilities come in handy. While "phrase match" is the most rigorous one, "match" is a better fit for the purpose of this prototype, while the user only needs to type in a term to get the wanted results back.

5.7 Integration of ElasticSearch and GraphQL into one Web Application: React

Since the separate parts of the solution have been discussed in the previous sections, this section describes how the ElasticSearch storage and GraphQL querying are integrated. This takes place by using the React web app that brings together 2 separate components and links them depending on what field from the results table is clicked. Following the previous example, where the text search input was "book", the query yields 6 results. If the user clicks on the Product description result, for example "Novelty book", then the correspondent field, "Description", from the OrderLine entity is highlighted.

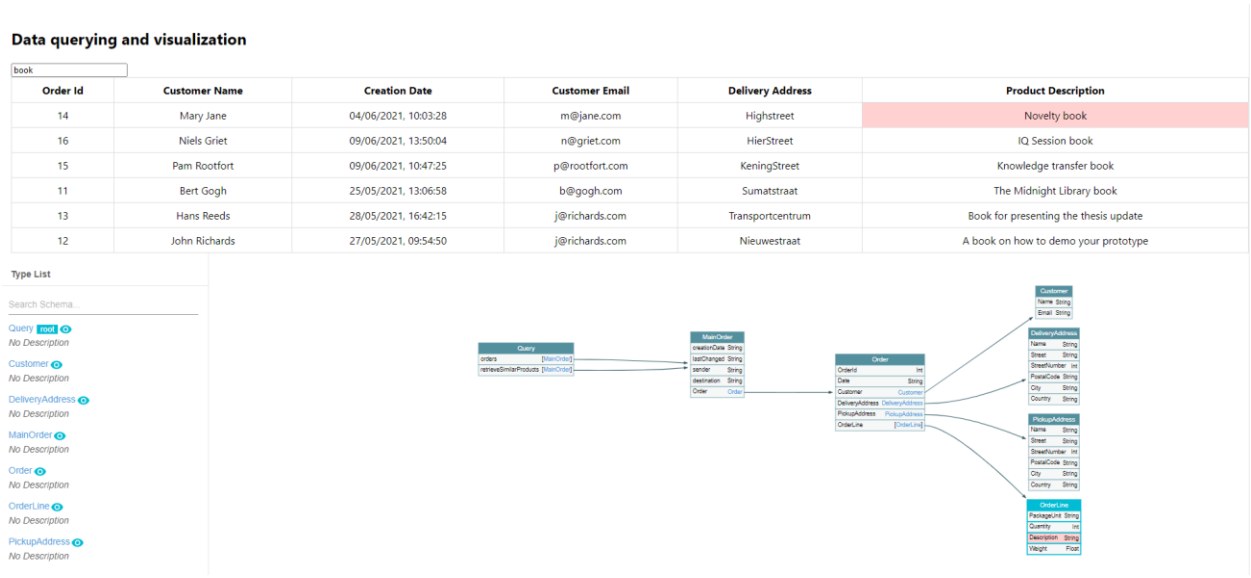


Figure 5.11 React app containing ElasticSearch results and schema visualization provided by GraphQL

The advantage provided by this powerful technology stack is that the correlation between the result and the schema can be clearly seen in one web application. Furthermore, an overview picture of the entities and their relations will make it clear where the entity is located within the whole schema. By clicking a specific result, its data type can also be identified. In our example, the Description field from the OrderLine entity has the data type “String”.

The example schema used for the prototype has a limited number of entities, but even if the schema will incorporate numerous entities, the visualization will not be hindered, because there is a Zoom in – Zoom out option to help focus on what is important.

The final solution for the prototype incorporates multiple functions that enable the user to browse through data, search for a specific order, visualize how the information is stored with the help of the data visualization tool and discover similar orders based on the description of a product.

6. Validation and Evaluation

Chapter 6 is meant to evaluate the implementation of the initial requirements, the perceived usefulness, ease of use and completeness of the developed prototype by the means of a survey distributed among experts within the collaborative company and discuss the results. The chapter addresses research question 5, *"To what extent is the designed prototype contributing to the goals of the stakeholders?"* and corresponds to the Implementation evaluation from the design cycle of Wieringa [44].

6.1 Technique: Single-case Mechanism Experiment

The single-case mechanism experiment technique was used for the validation step. It represents a test of a mechanism in a single object of study with a known architecture, which is presented in chapter 4. The research goal is to describe and explain the cause-effect behavior of the object of study. This can be used in implementation evaluation and problem investigation, where real-world research is done. It can also be used in validation research, where the validation models are tested. [44]

After the implementation of the proposed artifact, which is in fact a validation model, the next step is to evaluate it by using the questionnaire added as Appendix B to this document. The survey has 10 questions in total, both open and closed, and aims to evaluate the perceived usefulness, ease of use and completeness of the developed prototype, as determinants of user acceptance. Davis [53] defines the *perceived usefulness* as "the degree to which a person believes that using a particular system would enhance his or her job performance" and the *perceived ease of use* as "the degree to which a person believes that using a particular system would be free of effort." Taking this into consideration, the survey includes questions to incorporate both measures.

After the study executed by Davis [53], it turns out that users are driven to adopt an application primarily because of the functions it performs for them, and secondarily for how easy or hard it is to get the system to perform those functions. Furthermore, it should be emphasized that perceived usefulness and ease of use are people's subjective appraisal of performance and effort, respectively, and do not necessarily reflect objective reality.

When comparing two theoretical models for user acceptance, Davis [54] asked interviewees to list separately advantages, disadvantages and anything else related to the system evaluated. This approach was adopted as well with the survey for the "Data Observability prototype". From the same study [54], it turns out that designers believe that the key barrier to user acceptance is the lack of user friendliness, and that adding user interfaces that increase usability is the key to success. Data indicates that, although ease of use is clearly important, the usefulness of the system is even more important and should not be overlooked. Therefore, the questions in the survey aim to evaluate specific features of the prototype, based on the initial requirements, in order to establish their usefulness for users.

A full demo of the Orders use case was given to the development team of eMagiz and business consultants of CAPE Groep, in the role of experts. The functionalities that were explained and displayed are:

- the eMagiz messaging bus running locally and transporting Order messages from System A to System B;
- the data is stored in the ElasticSearch database, while the GraphQL server is up and running;
- the React app provides the UI for querying the results and schema visualization features, broadly explained in chapter 5;

The demo was followed by a Q&A session, where the participants got the chance to clarify any uncertainties surrounding the functionalities of the prototype. Afterwards, they were asked to fill in the evaluation survey, where an assortment of open and closed questions was used to better determine the perceived usability and completeness of the artifact.

6.2 Results Analysis

There are a total 8 answers registered and the mutual feeling received from the open questions regarding the prototype is a positive one, the functionalities presented being perceived as innovative for eMagiz and holding a lot of potential. The majority of respondents would consider using the solution as part of their work, because it seems intuitive and simple from a user perspective. There is also room for improvement, which will be discussed in chapter 7 with conclusions and future work.

The main purpose of the prototype is to create a data management solution by using the combined technologies of ElasticSearch and GraphQL, while using the eMagiz platform as starting point for saving information flowing in the integration IPaaS. For this purpose, the proof-of-concept prototype is demonstrated, covering the business case for a potential Transport Company, where Orders, Addresses and Customers are stored.

On one hand, the main positive aspects of the prototype, identified by the respondents, revolve around seeing the relation between the search results and the position of the specific fields in the visual data model which contributes to easily understanding the data, along with fast and easy searching across all fields.

On the other hand, among the negative aspects related to the prototype are the increased complexity if the visual model is too large and the fact that the prototype could be more dynamic and generic.

For the questions that used the linear scale from 1 (Not at all) and 5 (Completely), the table below presents the average and standard deviation per question:

No.	Question	Average	Standard Deviation
1	To which extent do you think the storing of metadata functionality was achieved? Currently, the creation date, last changed date, source and destination systems are saved.	4.125	0.64
2	To which extent do you think the text search option achieves the querying functionality?	3.75	1.03

3	To which extent do you think the schema visualizer achieves the navigation and visualization functionality?	4.25	0.46
4	To which extent do you think the connection between visualizing the data and its schema has been achieved? (keeping in mind the highlighting of the fields)	4.25	0.7
5	To which extend do you think the prototype is easy to use?	3.875	0.64
6	To which extend do you think the prototype is complete?	3.375	0.74
7	Please provide a general score for the whole solution.	4	0.63

Table 4.1 Analysis of linear scale survey questions with average and standard deviation

To start with the general score for the whole solution (question 7 from the above table), an average 4 out of 5 is a positive feedback. For questions 4 and 3, an average score of 4.25 was returned, which means the schema visualizer and the connection between the search results and the schema by highlighting the fields successfully achieved the navigation and data connection features. Question 1, with the average score of 4.125, shows that storing the metadata was favorably implemented.

Shifting towards the lower rated aspects (below a score a 4), question 5 has an average of 3.875, which shows that the prototype does not seem to be easy to use from the perspective of some experts. To tackle that, the prototype incorporates tooltips for the various sections, to describe the features. Question 2, with the average of 3.75, shows that the text search option did not reach its full potential for the query functionality, in the eyes of some respondents. Finally, question 6 holds the lowest average score at 3.375, regarding the completeness of the prototype. As an add-on for this, when the survey was distributed among the respondents, the prototype was still undergoing changes and some more features were added afterwards, such as the related products functionality.

7. Conclusion

This final chapter discusses the findings of the research by firstly answering each research question proposed, then showing the scientific as well as the practical contributions brought. The limitations, future work and recommendations for practice are addressed likewise.

7.1 Research Questions

This master project proposed one main research question:

"How to design an efficient data management retrieval and reporting solution for client data based on ElasticSearch and GraphQL?"

This was split into 5 sub-questions which will be discussed and answered in this section.

RQ1 "What kind of data management architecture can be used for retrieval and reporting of client data?"

In order to be able to respond to this inquiry, a systematic literature review was conducted to investigate the scientific publications on existing data management architectures for retrieval and reporting of client data, broadly discussed in Chapter 2. The literature review is split into 2 parts. In the first part, several non-relational databases were analyzed and classified, while discussing their advantages and disadvantages. ElasticSearch represents our point of interest because of its search engine performance and its ability to store unstructured data, therefore the architecture proposed by this master project is built while using it for data storage. A comparison between relational and non-relational databases is shown, in order to determine which system would be suitable considering the needs of the company that adopts it. The second part of the literature review focuses more on data governance and data quality, providing several frameworks, methodologies and metadata architectures, along with use case examples. Data lakes are explained, including data cleaning and dataset discovery that are linked to them.

RQ2 "What are the needs of the business users regarding this data management architecture?"

This question is broadly answered in Chapter 3, Requirements. A business user was not identified to implement the prototype within the business clients of the collaborative company, eMagiz. Therefore, a survey was distributed among the Chief Technology Officer, the Software Delivery Manager, the Product Owner, a Software Developer, and a Business Consultant in the role of experts, in order to find the requirements base for the prototype developed as a proof of concept. There are 3 personas identified that will interact with the artifact and that have different use case scenarios and expectations: the business user from a logistics company, the architect within eMagiz and a data scientist that can be hired for further analysis of the gathered data. There are 11 user stories focusing on functional and non-functional requirements. The first category of requirements (functional) mainly addresses the storing, querying, visualizing and navigating through the data saved while using the eMagiz platform. The second category of requirements (non-functional) are targeting the speed (latency), security, reliability and

maintainability of the solution. The prototype might be used by the development team for internal purposes as well, such as testing different flows and analyzing the data and metadata saved with different integrations.

RQ3 “How to design a data management architecture based on ElasticSearch and GraphQL for retrieval and reporting of client data?”

This question is broadly discussed in chapter 4, that proposes an architecture specification model constructed by using the Archimate modelling language. The architecture is built upon the existing infrastructure from eMagiz and proposes an extension on the technical layer with the ElasticSearch and GraphQL servers, the first one being used for storing the data and the latter for building up queries and visualizing the data schema. Additionally, there are specific components that are added in the application layer, such as the “Process data service” and the “Data processing function” to support the new functionalities proposed by the “Data Observability prototype”, such as storing the data and making it available for the user to visualize it. On the business layer, the addition is the appended business processes for “Defining settings for storing data and metadata” that corresponds to the architect role, the “Extract and process data” related to the data scientist role and the “Explore results for taking decisions” linked to the customer role. The newly added elements have a blue font, so they can be easily identified. The architecture specification model gives an overview on the entire design for the prototype and shows how the components are operating as a whole, as part of the eMagiz existing architecture.

RQ4 “How to integrate the data management architecture in the business context of a Dutch organisation?”

The integration of the data management architecture in the context of eMagiz is widely explained in chapter 5, showing the implementation of the prototype, based on a use case for a logistics company, saving information about Orders. eMagiz is the starting point, having the integration platform that facilitates the transport of messages between systems and applications. Before, the messages would either reside at the source or at the destination system and were not saved in between. The prototype proposes to intercept the original message sent, enhance it with metadata (such as timestamp), and store it in the ElasticSearch database. The User Interface is provided by a web application developed in React, where data can be searched and analyzed. By using GraphQL, the data schema can be visualized along with the associations between entities and the information residing in them can be browsed. GraphQL also supports a feature of finding related data based on text search, which gives the data management architecture a new specialty.

A wireframe is created to present a possible configuration that could be integrated into the eMagiz platform for saving a specific type of message that is used for an integration. Due to the fact that the User Interface is built with React, which is based on javascript, it could be easily incorporated in eMagiz as an extra page in the last lifecycle step, “Manage”.

RQ5 “To what extent is the designed prototype contributing to the goals of the stakeholders?”

Chapter 6 projects the validation and evaluation step after the implementation of the prototype. A survey is distributed among various experts in the field from eMagiz and CAPE

Groep, having the role of software developers and business consultants. They were shown a demonstration of the functionalities of the prototype before being asked to fill in their answers. The survey consists of several open and closed questions that aim to evaluate the perceived ease of use and completeness with regards to the initial requirements and whether there would be further interest for the company in using the prototype. The models of Davis [53], [54] and Venkatesh [55] were used as guidance when setting up the survey questions. There are 8 answers gathered and the general score provided by the respondents is 4 out of 5, which shows an optimistic future for the prototype. The highlighting between the search results and the data schema for a specific field is the most valuable feature in the eyes of the respondents.

7.2 Contributions

This Master project presents the findings of a systematic literature review for data management platforms, proposes an architecture specification model based on ElasticSearch and GraphQL and discusses the implementation of the prototype within the collaborative company eMagiz. In the following sections, the scientific as well as the practical contributions are shown.

7.2.1 Scientific Contributions

The scientific contributions brought by this master project revolve around the exhaustive systematic literature review on data management platforms, which provides an overview for researchers interested in data governance and its applications. The master project also includes a design of a new architectural approach including the technologies of ElasticSearch and GraphQL within the collaborative organization eMagiz (Chapter 4), along with the implementation part of the designed prototype (Chapter 5). No scientific publications were found during the literature review to include the above-mentioned technology stack; therefore, this knowledge gap was filled and contributes to the novelty brought by the project.

During the problem treatment design phase, a survey was shared among experts in the field of application integrations to discover the requirements for a data management tool that could store data, search and visualize the data schema. This can be viewed as a starting point when designing a data management architecture.

7.2.2 Practical Contributions

The major practical contribution resides in the design and implementation of the prototype that has a use case for saving Orders and represents a data management platform. The prototype has as starting point the eMagiz integration platform, where messages that are sent between systems and applications are stored, with ElasticSearch playring the role of the database. A user can interact with the data by using the React web application, which provides searching and schema visualization features, supported by a GraphQL server. Being able to visualize the data that is flowing within the platform helps for debugging and tracing purposes. The prototype is a data management platform which stores information and includes a search function to look through the set of data and find a specific record.

This master project provides a guide on the design and implementation of a data management platform as a tangible example and opens new research business possibilities for the collaborative company.

7.3 Limitations

As for any short-term projects, there were some limitations that occurred regarding the master assignment and are shown in the section below. The most noticeable one is the scarcity of existing research on the topic for data management platforms by using the specific technology stack of ElasticSearch and GraphQL. Another limitation was the absence of a business client to provide a real-life case for the implementation part. Another downside, related to the previous one, is that validation could not take place with end-users. Last but not least, the time restriction made the project to be developed up to a certain level, in accordance with the time that was available.

Firstly, the scarcity of existing research for the specific topic chosen with regards to the preferred technology will be discussed. As presented in chapter 2, which provides the systematic literature review to aid with background information for data management architectures and platforms, there is no existing solution to include the technology stack of ElasticSearch and GraphQL, because this is highly specific and tightly connected to the implementation segment. On the other hand, there are frameworks and architectures presented that come as guidance and provide theoretical concepts for implementing such a platform and different solutions are reviewed and explained.

The second limitation mentioned is the absence of a business client to provide a real-life case for implementing the prototype. To combat this restraint, the solution is developed to use a scenario typical for a logistics company, to process mock order data. This can prove convenient for eMagiz, as most of the clients are related to this business segment and the example can appeal to them, being easy to understand and to relate to. The prototype is therefore static and build around the instance mentioned, but it provides a functional solution example for a data management platform. The prototype can act as a standalone project, not being necessarily connected to eMagiz, as long as another source that provides data is taken into account.

The last limitation identified regards the impossibility of validating the prototype with end users due to missing a business client. Therefore, the validation and evaluation take place with experts from the application integration field within eMagiz and CAPE Groep, both software developers and business consultants, and they deem the prototype as holding a lot of potential. This represents a qualitative validation, but it might seem incomplete as the experts are part of the same organization and bias could occur in this case.

7.4 Future Research

Once this research project was completed, new possibilities were discovered that could be suitable for future research. First, to tackle the limitation of narrow validation within one company, this step could be extended with several different parties to make sure the results are not biased and to analyze the opinion of more experts on the matter. Additionally, more dimensions can be investigated, and the survey can be improved to assess more aspects of the

prototype in different setups. The UTAUT method by Venkatesh [55] can be used to extend the questionnaire. For example, questions regarding the decrease of time needed for the important job responsibilities or questions that asses the increase of effectiveness of performing job tasks can be included in the survey. Regarding outcome expectations, questions related to increasing the quality of the output of the job can be considered. The negative impact on users can also be evaluated, such as the complexity of the prototype, or if users perceive it as too complicated. Another aspect to consider is the compatibility, whether users think the system into their work style or not. Nevertheless, a major value addition would be brought to the solution if a real-life case would be found and implemented for the data management platform, as currently the prototype is in an experimental form.

Second, besides validating the functionalities of the solution proposed, the architecture model solution could also be exposed to validation for external validity, and the model could be changed so that it is not embedded into one particular platform, such as eMagiz, making it more valuable and generic for other use cases as well. Nevertheless, the architecture model provides a good starting point for researchers interested in the topic discussed and adds to the general knowledge base.

7.5 Recommendations for Practice

As a series of limitations and future research possibilities were identified during this master project, naturally the recommendations for practice will include them as well. One downside discussed is the fact that the prototype is static, therefore more work will have to be devoted to turn it into a general solution that could adapt to the specific needs of a client. An important advantage is that eMagiz uses the CDM (Canonical Data Model) that contains the data schema, therefore it could be used for setting up the GraphQL server, which needs a clear definition of the data structure. Nevertheless, the prototype developed provides a starting point and an example of how a data management platform can function based on ElasticSearch and GraphQL, while being linked to the eMagiz IPaaS.

Further development is also needed for incorporating the prototype into the eMagiz platform, so that users do not have to use a different application (React) to search through the data. This can be achieved since the web application is based on javascript, which makes the functionalities to be easily transferrable into the eMagiz solution as widgets. If the data management platform will be part of the last lifecycle step, “Manage”, then users will have a complete experience, where they can interact with the information that is flowing from the integrations, being able to browse it and find related data. The most difficult part will be to find a business client that is interested in such a product and have an applicable business case, backed by a user journey that reflects the user scenarios where pain points are addressed.

The “Data Observability prototype” has features that can be further extended and improved with regards to security and user experience to accommodate more elaborate needs, and to become a unified eMagiz solution. For example, with regards to the user interface, the prototype is built into one single web page that includes all the functionalities. This could be broken down into separate sections or tabs, for a focused experience. Regarding the security, there is no

login function in place for the prototype, apart from the eMagiz login used when transporting the data with the messaging bus, since the solution is meant to be integrated into the platform.

The metadata (timestamp, last changed, source and destination) that is saved along with the original message can be explored further, because currently only the timestamp is used for sorting the results starting with the most recent one after a search is done.

Appendix A

Expert opinion on requirements definition

The survey was distributed in the collaborative company, eMagiz, among the Chief Technology Officer, the Software Delivery Manager, the Product Owner, a Software Developer for the platform, and a Business Consultant.

Question 1: The main intended functional features of this solution are to store business data while it is being transported between systems/applications, and then query it inside the eMagiz platform. Do you think such a functionality would raise interest among eMagiz business users and would prove to be useful? Can you think of a business case example for it?

Responses:

- Yes it would be useful and valuable, because users have the ability to search and find their business data. It could be useful when you quickly want to know the values of your master data (CRM information or Employee information). It could also be useful when you need urgently and dynamically need data from your business critical processes.
- Querying it inside the platform like that would mainly benefit eMagiz users during testing, debugging or doing a root cause analysis. Particularly in cases where data went "missing" somewhere in their landscape, this could help customers to more quickly find the cause and fix the issue.
- Depends on why and what you want to store. If it is making the work of customers more efficient during setting up these solutions because querying data is easy....YES. However there is a risk in storing data, you need to know exactly what you want to store, what you may store, how long you want or must store it and of course what is not of interest for the solution to store to make querying even possible. A good business case I would say something that we currently are doing with metrics data from a integration pattern... Having a layer between source and customer to create even more specific overviews of their metrics in their own landscape with using this graphQL api would be really nice. Extra set of endpoints in our api gateway :)
- I think the best way to find this out would be to ask the business users. I think this would have value, given that the dataset is complete. As a business user, I would want a data hub where I can query my data with some guarantees, and not just hope that the data is there because it crossed some integration for some reason. One possible use case could be: Find all users that have placed an order in the last month. One possible use case that you maybe couldn't support is: Find all users (because they must be known by eMagiz, which is maybe not always the case if the system already exists before integration, etc).
- One case I can think of where business users could be interested in this is the case of an insurance company. In an application and integration landscape for insurance

companies based on a microservices architecture multiple systems are responsible for different parts of the process. These systems are therefor also the owners of different parts of the data used for the process. When using an integration platform to store the different data bits of the different processes, this platform can be a suitable solution for visualizing this data from different systems and processes to be able to get fast insights in your data. A concrete example for this could to quickly find related insurance cases based on the characteristics of a claim / case.

Question 2: If a business client would express their desire to have this storing and querying functionality enabled, how do you think this should be integrated into the eMagiz platform? (on what step from the lifecycle, create extra screens or reuse of the existing ones?)
Responses:

- I would like to see the data in Manage, because this is the place where I can see my environments. I would expect to see a visual representation of the data and its structure, so reusing one of the already built widgets (CDM and schema widgets) would be advised.
- Definitely in the Design phase, because other decisions that have an impact on hosting/licensing costs and information security are also taken there. And this needs to be decided before you start building/deploying anything (Create/Deploy phases). And we probably want the user to select which integrations (or message types?) to store or not store.
- When its development related querying the complete CDM would be a improvement. When its testing related in the create phase as addition to the "flow testing" when it's an additional feature thing it should be in deploy as additional endpoints in the gateway that can be visualized/used in the manage monitoring overview.
- Storing could be simple, like a checkmark in Design on the integration edit screen, and then optionally provide extra fields like the schema, retention, etc. Querying would be something for the manage phase (or perhaps a new "insight" phase later). Could be one screen in the beginning for querying and analytics, but add more later.
- The manage phase would probably be the best phase to add this new functionality. Otherwise a new phase called "Vizualize" could also be an idea. I would think that partly reusing some of the screens that are already there could be a good idea. A visual model could be created based on the CDM data model from design and perhaps it could be a good idea to add "tags" to entities to illustrate to the business user from which system the data originates.

Question 3: If metadata will also be stored along with business data, what kind of metadata do you think should be considered?

Responses:

- Time, originating system, target system, any user data available (i.e. for API gateway), message type, whether it's an error or not.
- I want to query attributes, related entities (relations) and also related metadata (like last changed and timestamp).

- Timing (when did the data go through eMagiz) and routing (where did the data come from and where did it end up).
- Sizes, amounts, dates, versions, permissions.
- At least the latest update datetime for an entity, also from which system the data originates and if possible to which system(s) the data flows.

Question 4: Besides the storing and querying functional requirements described above, do you think there are other prerequisites(functionalities) worth considering to be included, while being related to the scope of the assignment?

Responses:

- I'd say keep the scope absolutely minimal and think about fully functional store and query before thinking about more things. Then, you could make graphs with statistics about the stored data, nice widgets for showing the data, etc. At some time, if we really want this to be a data hub, perhaps we also need functions to import data that's not already flowing through eMagiz to get complete view of data.
- Not clear what you mean but a playground around the graphQL would be the bomb. as in GraphQL playground is an external interactive editor for your GraphQL queries. It is based on GraphQL and accessible through the web browser. see Example Saleor Commerce.
- Giving users access to the data directly (through GraphQL) and not only with a UI in our platform could add a big business case: reporting on their own data using whatever BI tooling (that supports GraphQL) they prefer.
- Exporting the data in another format Building quickly an API on it and make it accessible for others Show relations between entities, showing the graph view.
- Add security to the data entities because not all users will probably be allowed to view all the data that is being stored.

Question 5: Switching to the non-functional category of requirements, what should definitely be taken into account for this assignment? For example security concerns, accessibility, efficiency.

Responses:

- Both the performance and costs are directly related to the amount of data stored. In a messaging solution where we constantly receive new messages, this would probably mean having a relatively short retention policy. But the shorter it is, the less value the feature offers... Security is also an attention point: normally eMagiz delivers data to connected applications and those applications are responsible for enforcing the correct data access privileges to their users. If eMagiz starts storing data and making it available to users, we also get that responsibility.
- Search should take maximum 1 second (performance) Language of the data should also be known.
- Security, speed (latency), reliability.
- Data security.

- I think that security is definitely something to consider, but not as a non-functional. If you're going to store business critical data, some security should be in place. Another non functional could be that business users should be able to only use the data visualization part of the platform and not be "bothered" with the other phases in the platform.

Question 6: Following a typical eMagiz approach, what would be the gains(benefits) for implementing the features described in the assignment?

Responses:

- Attract new users and retain existing users by being the 'platform' that can do everything integration related. Reduce dependency, and complexity, of using and setting up other tools for seeing what flows through your integration.
- Make it small as possible, pick a pattern, talk to customers to feel their pain and give them something as this which is than making their daily routine so much better.
- Users can explicitly see the data they store, normally they could never see the data in eMagiz, but only in source or destination systems.
- See business case questions.
- It could help to close a part of the gap between consultants/developers and the business with regards to data. Data is often shown in a user friendly format, but by exposing the underlying data to business users it could be possible to create a better understanding for the business users and speed up the development process for new functionality.

Question 7: What would be the pain points(difficulties) of implementing the features described in the assignment?

Responses:

- I'm not sure about technical details, but I think schema conformance of the data that's passed through. I think GraphQL is highly schema dependent so this should work smoothly, with schema and field detection, etc, so you can query all data without having the user specify in detail the structure and schema of his messages.
- Data syncing (how do you know this data is not outdated) How long do you store it How to make it secure How to give easy access.
- Too many angles on what data is of importance for businesses that are operating and storing/retrieving this with eMagiz.
- As mentioned two questions up, performance and security.
- Probably the security of the data. You should be able to ensure to users that the data that are being stored are stored securely. Also I'm not sure how easy it is to develop custom queries based on business user need, but I think it should be easy for them to create new queries at wish.

Appendix B

Expert opinion on the prototype validation

A full demo has been given to the eMagiz team on the functionalities of the prototype developed, following the workflow:

- the eMagiz messaging bus running locally and transporting Order messages from System A to System B;
- the data is stored in the ElasticSearch database, while the GraphQL server is running;
- the React app provides the UI for the querying and schema visualization features;

After the demo took place, the software developers, business consultants and the Development Manager were asked to fill in anonymously the survey below:

Question 1: Would you consider using this prototype for your own work?

Responses:

- Yes.
- Yes, I would like to use it during development to check how my data is changed during the workflow.
- Yes.
- I don't build integration solutions in eMagiz myself, so for my "own work" the answer would be no.
- Yes, we are currently working on enterprise archiving functionality using Amazon S3. This has strong limitations with regard to search (you can only search by file name) and explorability, also from a performance perspective. I see strong relations between the prototoype and archiving (wiretapping flows to send it to a data storage and then querying this to get results) and I think the design presented could be a future proof, and more dynamic replacement for the S3 archiving functionality that is currently offered in eMagiz.
- I think we do have to overcome some obstacles with this prototype, like automatically inferring the schema from the CDM to make this work in any customer environment, to prevent manual creation of schema's. But this is a great PoC to demonstrate the capabilities of GraphQL and Elastic for Archiving and with clear next research steps for going from a PoC to a MVP used in enterprise production, I would certainly consider implementing this in eMagiz.
- Yes, after developing something with elastic search, I think it could be of use to visualise how I used elastic search data.
- I believe it holds value for larger customers that want the ability to derive more context from the data models that are captured within eMagiz. For myself in my role as Expert Services I don't directly see how this prototype will help me with my daily work since a

big chunck of the prototype revolves around informing an end user on the data content (and I as an outside user should not be able to see that with ease).

Question 2: To which extent do you think the storing of metadata functionality was achieved? Currently, the creation date, last changed date, source and destination systems are saved.

Responses:

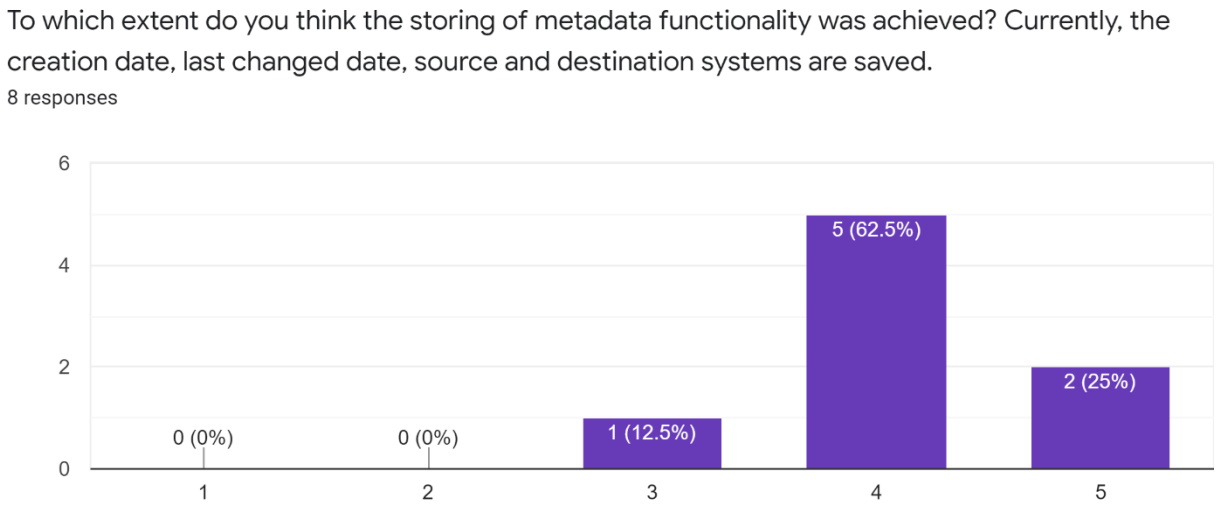


Figure B1 Results to Validation Survey question 2

Question 3: To which extent do you think the text search option achieves the querying functionality?

Responses:

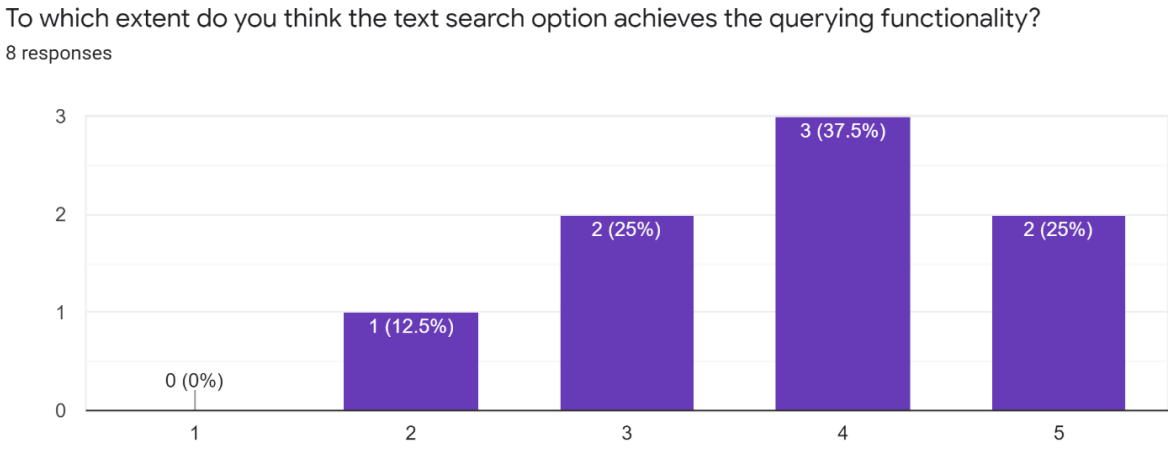


Figure B2 Results to Validation Survey question 3

Question 4: To which extent do you think the schema visualizer achieves the navigation and visualization functionality?

Responses:

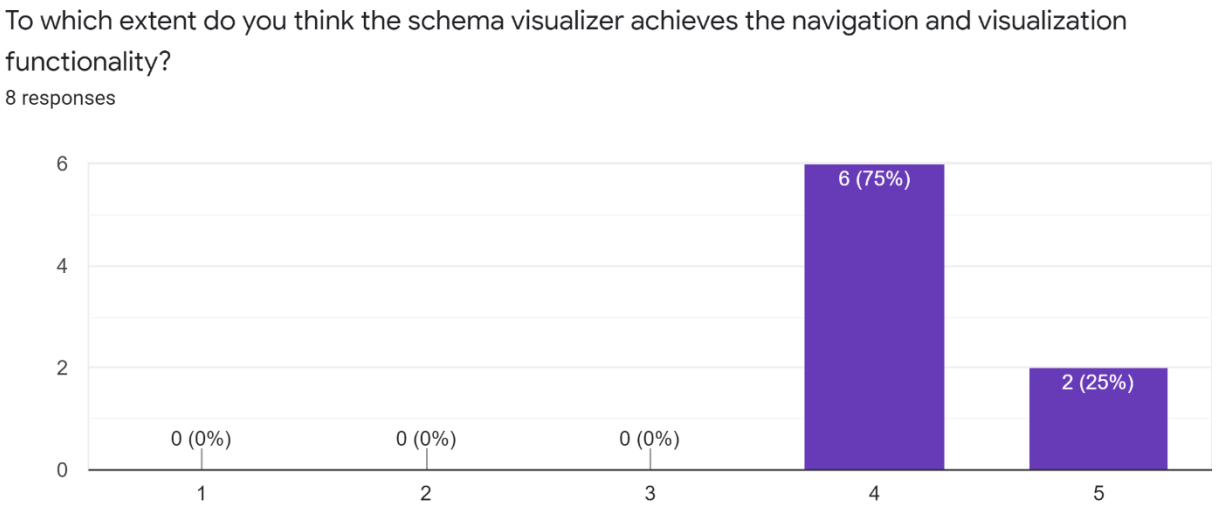


Figure B3 Results to Validation Survey question 4

Question 5: To which extent do you think the connection between visualizing the data and its schema has been achieved? (keeping in mind the highlighting of the fields)

Responses:

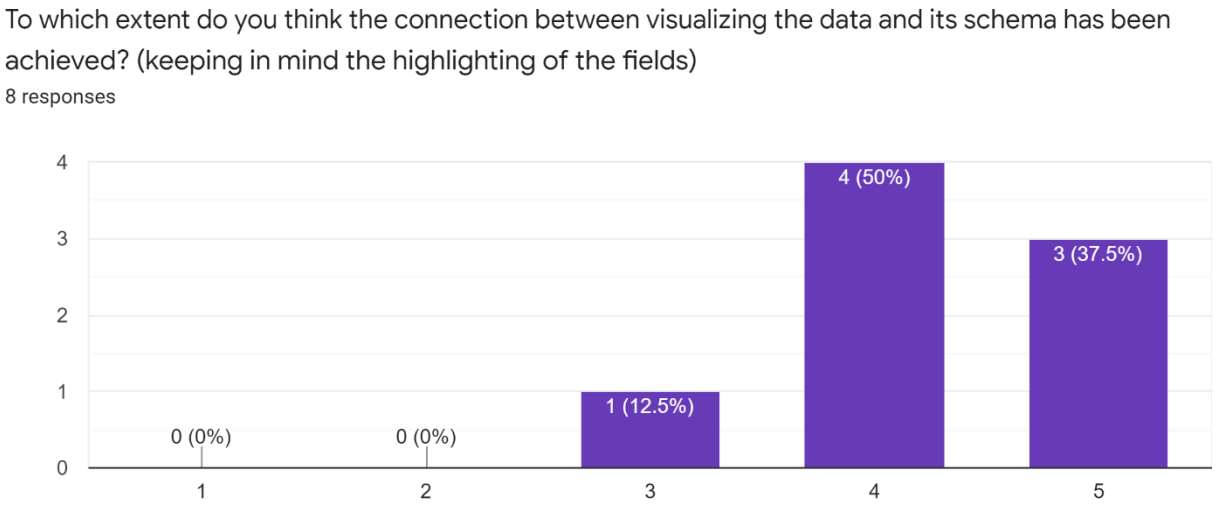


Figure B4 Results to Validation Survey question 5

Question 6: To which extend do you think the prototype is easy to use?

Responses:

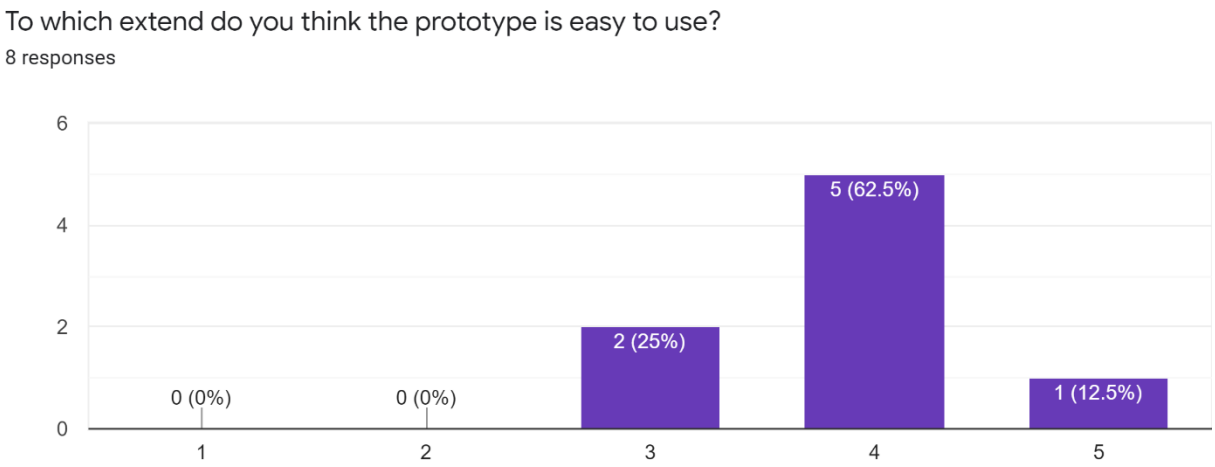


Figure B5 Results to Validation Survey question 6

Question 7: To which extend do you think the prototype is complete?

Responses:

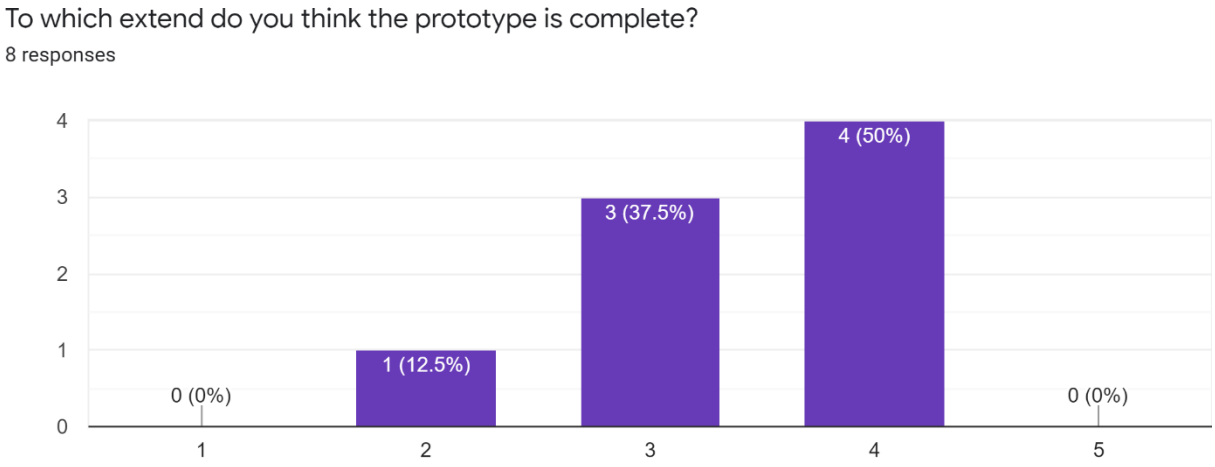


Figure B6 Results to Validation Survey question 7

Question 8: What are the positive aspects that you see in the prototype?

Responses:

- See the relation and position of data in the schema;
- Prototype result is innovative and new, we never had this functionality.
- Easy searching, clear highlighting of the selected data.
- Easy to see how this would "automagically" (from a user perspective) work for any data model in eMagiz. Nice visualization of the data structure in Elasticsearch that is very recognizable for eMagiz users.
- As far as I can see it has a lot of complexity going on with several systems that are dependent on each other yet it feels intuitive and simple from a user perspective.
- easy to navigate the structure of data.
- The prototype appears to be lightning fast in searching, and I would also expect this to scale well based on the architecture presented. Also, search can happen visually and search happens over all fields. I think this is very desired functionality (given that we would also offer filter options in the future). The ability to see search results in the context of the visual model and show the relations between them is great for understandability.

Question 9: What are the negative aspects that you see in the prototype?

Responses:

- Too many things if the schema is complex;
- You need a lot of context before you can use it probably;
- Not really "negative", but perhaps it could be possible to also visualize (highlight) which result you have selected to be visualized in the schema;
- While the data structure is shown very nicely, I'm not sure how the table-based search results would work in a complex data model with a lot of data. Also, while the demo showed how to store metadata, it's unclear to me you'd effectively use this metadata in practice.
- needs more polish;
- This prototype has been built for a specific schema, effort would need to be made to make this generic and apply it to any customer environment. Naturally, this can't be expected within the scope of the current research (I think already a lot is done within limited time) but nevertheless its an obstacle to overcome in the future during implementation in production.
- I think the tool could use some more documentation or tooltips for the user, so that the user truly knows the strengths and relevance of the tool!
- You need a lot of context before you can use it probably;

Question 10: Please provide a general score for the whole solution.

Responses:

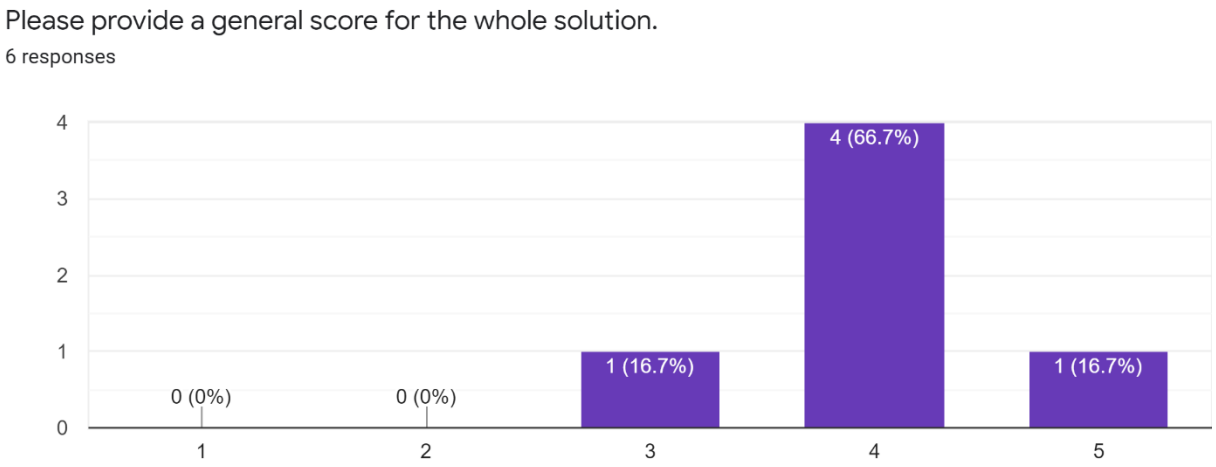


Figure B7 Results to Validation Survey question 10

References

[1] Towards Data Science, *The difference between available and data and accessible data*, https://towardsdatascience.com/the-difference-between-available-data-and-accessible-data-638ed3dc538b, visited on 11.01.2021

[2] Harrington, Jan L. *Relational database design and implementation*. Morgan Kaufmann, 2016.

[3] Vicknair, Chad, et al. *A comparison of a graph database and a relational database: a data provenance perspective.* Proceedings of the 48th annual Southeast regional conference, 2010.

[4] Gyorödi, Cornelia, Robert Gyorödi, and Roxana Sotoc. *A comparative study of relational and non-relational database models in a Web-based application*. International Journal of Advanced Computer Science and Applications 6.11 (2015): 78-83.

[5] Pramod Sadalage, *NoSQL Databases: An Overview*, http://www.thoughtworks.com/insights/blog/nosql-databases-overview, visited on 25.01.2021

[6] Băzăr, Cristina, and Cosmin Sebastian Iosif. *The transition from rdbms to nosql. A comparative analysis of three popular non-relational solutions: Cassandra, mongodb and couchbase*. Database Systems Journal 5.2 (2014): 49-59.

[7] Jatana, Nishtha, et al. *A survey and comparison of relational and non-relational database.* International Journal of Engineering Research & Technology 1.6 (2012): 1-5.

[8] Okoli, Chitu, and Kira Schabram. *A guide to conducting a systematic literature review of information systems research* (2010).

[9] Kononenko, Oleksii, et al. *Mining modern repositories with elasticsearch*. Proceedings of the 11th working conference on mining software repositories. 2014.

[10] Čerešňák, Roman, and Michal Kvet. *Comparison of query performance in relational and non-relational databases.* Transportation Research Procedia 40 (2019): 170-177.

[11] Venkatraman, Sitalakshmi, et al. *SQL versus NoSQL movement with big data analytics*. Int. J. Inform. Technol. Comput. Sci 8 (2016): 59-66.

[12] Acharya, Biswaranjan, et al. *NoSQL Database Classification: New Era of Databases for Big Data*. International Journal of Knowledge-Based Organizations (IJKBO) 9.1 (2019): 50-65.

[13] Vokorokos, Liberios, Matúš Uchnár, and Lubor Leščišin. *Performance optimization of applications based on non-relational databases*. 2016 International Conference on Emerging eLearning Technologies and Applications (ICETA). IEEE, 2016.

[14] Greca, Silvana, Anxhela Kosta, and Suela Maxhelaku. *Optimizing Data Retrieval by Using Mongodb with Elasticsearch*. RTA-CSIT. 2018.

[15] Bhatt, Preeti. *Performance Comparison between Column Store NoSQL Databases.*, International Journal of New Innovations in Engineering and Technology, 2018

[16] Khazaei, Hamzeh, et al. *How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey*. Big Data & Information Analytics 1.2&3 (2016): 185.

[17] Mami, Mohamed Nadjib, et al. *Squerall: Virtual ontology-based access to heterogeneous and large data sources*. International Semantic Web Conference. Springer, Cham, 2019.

[18] Bondiombouy, Carlyna, and Patrick Valduriez. *Query processing in multistore systems: an overview.* International Journal of Cloud Computing 5.4 (2016): 309-346.

[19] Mehmood, Erum, and Tayyaba Anees. *Performance analysis of not only SQL semi-stream join using MongoDB for real-time data warehousing*. IEEE Access 7 (2019): 134215-134225.

[20] Asaad, Chaimae, Karim Baïna, and Mounir Ghogho. *NoSQL databases: yearning for disambiguation.* arXiv preprint arXiv:2003.04074 (2020).
[21] Brewer, E. A. (2000, July). *Towards robust distributed systems*. In PODC (Vol. 7).
[22] Mami, Mohamed Nadjib, et al. *The query translation landscape: a survey*. arXiv preprint arXiv:1910.03118 (2019).
[23] Mami, Mohamed Nadjib, et al. *Querying data lakes using spark and presto*. The World Wide Web Conference. 2019.
[24] Olivier Curé, Robin Hecht, Chan Le Duc, and Myriam Lamolle. 2011. *Data integration over nosql stores using access path based mappings*. In International Conference on Database and Expert Systems Applications. Springer, 481–495.
[25] Rami Sellami and Bruno Defude. 2018. *Complex queries optimization and evaluation over relational and NoSQL data stores in cloud environments*. IEEE Transactions on Big Data 4, 2 (2018), 217–230.
[26] Paolo Atzeni, Francesca Bugiotti, and Luca Rossi. 2012. U*niform access to nonrelational database systems: The SOS platform.* In International Conference on Advanced Information Systems Engineering. Springer, 160–174
[27] Boyan Kolev, Patrick Valduriez, Carlyna Bondiombouy, Ricardo Jimenez-Peris, Raquel Pau, and José Pereira. 2016. *CloudMdsQL: querying heterogeneous cloud data stores with a common language*. Distributed and parallel databases 34, 4 (2016), 463–503.
[28] Martin Giese, Ahmet Soylu, Guillermo Vega-Gorgojo, Arild Waaler, Peter Haase, Ernesto Jiménez-Ruiz, Davide Lanti, Martín Rezk, Guohui Xiao, Özgür Özçep, et al. 2015. *Optique: Zooming in on big data*. Computer 48, 3 (2015), 60–67
[29] Lu, Wen, Ligu Zhu, and Shufeng Duan. *Research and implementation of big data system of social media*. 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). IEEE, 2018.
[30] Zhang, Ning, and Q. Yuan. *An overview of data governance.* Economics Paper, December (2016).
[31] Priebe, Torsten, and Stefan Markus. *Business information modeling: A methodology for data-intensive projects, data science and big data governance*. 2015 IEEE International Conference on Big Data (Big Data). IEEE, 2015.
[32] M. Ferguson, *Building an Enterprise Data Reservoir and Data Refinary*, European TDWI Conference 2015, Munich, 2015.
[33] Foster, Kyle, et al. *Business intelligence competency center: Improving data and decisions.* Information Systems Management 32.3 (2015): 229-233.
[34] Attard, Judie, and Rob Brennan. *Challenges in value-driven data governance.* "OTM Confederated International Conferences. On the Move to Meaningful Internet Systems". Springer, Cham, 2018.
[35] Laney, D.: Gartner Analytic Ascendancy Model. Information Economics, *Big Data and the Art of the Possible with Analytics* p. 33 (2012), https://www-950.ibm.com/events/wwe/grp/grp037.nsf/vLookupPDFs/Gartner Doug- Analyt-ics/$le/Gartner Doug- Analytics.pdf
[36] Chen, Y.: *Information Valuation for Information Lifecycle Management*. In: Second International Conference on Autonomic Computing (ICAC'05). pp. 135{146. IEEE (jun 2005). https://doi.org/10.1109/ICAC.2005.35, http://ieeexplore.ieee.org/document/1498059/

[37] Wijnhoven, F., Amrit, C., Dietz, P.: *Value-Based File Retention*. Journal of Data and Information Quality 4(4), 1{17 (may 2014). https://doi.org/10.1145/2567656, http://dl.acm.org/citation.cfm?doid=2628135.2567656
[38] Stander, J.B.: The Modern Asset : Big Data and by (December) (2015)
[39] Nargesian, Fatemeh, et al. *Data lake management: challenges and opportunities*. Proceedings of the VLDB Endowment 12.12 (2019): 1986-1989.
[40] Dai, Wei, et al. *Data profiling technology of data governance regarding big data: review and rethinking.* Information Technology: New Generations (2016): 439-450.
[41] Kachaoui, Jabrane, and Abdessamad Belangour. *From single architectural design to a reference conceptual meta-model: an intelligent data lake for new data insights.* International Journal 8.4 (2020).
[42] DeStefano, R. J., Lixin Tao, and Keke Gai. *Improving data governance in large organizations through ontology and linked data.* 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). IEEE, 2016.
[43] Hype Cycle for Analytics and Business Intelligence, 2020, Gartner, https://www.gartner.com/en/documents/3988448/hype-cycle-for-analytics-and-business-intelligence-2020, visited on 08.04.2021
[44] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Berlin,Heidelberg: Springer Berlin Heidelberg, 2014.
[45] I. Alexander, A taxonomy of stakeholders: Human roles in systems development. Int. J. Technol. Human Interact. 1(1), 23–59 (2005)
[46] Archi archimate modelling, https://www.archimatetool.com/, visited on 05.05.2021
[47] The TOGAF standard, https://www.opengroup.org/togaf, visited on 05.05.2021
[48] Medeiros, Juliana, et al. *Requirements Engineering in Agile Projects: A Systematic Mapping based in Evidences of Industry*, CibSE. 2015.
[49] Williams, Clay, et al. *Supporting enterprise stakeholders in software projects,* Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering. 2010.
[50] eMagiz, The solution for all integration challenges, https://www.emagiz.com/, visited on 21.05.2021
[51] GraphQL-voyager, Interactive visualization of the graph data model and explore the API, https://github.com/APIs-guru/graphql-voyager
[52] Demystifying ElasticSearch Queries, https://qbox.io/blog/elasticsearch-queries-match-phrase-match/, visited on 27.05.2021
[53] Davis, Fred D. *Perceived usefulness, perceived ease of use, and user acceptance of information technology*. MIS quarterly (1989): 319-340.
[54] Davis, Fred D., Richard P. Bagozzi, and Paul R. Warshaw. *User acceptance of computer technology: A comparison of two theoretical models.* Management science 35.8 (1989): 982-1003.
[55] Venkatesh, Viswanath, et al. *User acceptance of information technology: Toward a unified view.* MIS quarterly (2003): 425-478.