People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

SAAD DAHLEB University - Blida 1
Faculty of Sciences
Department of Informatics

Graduation Project

To obtain the title of Master in Computer Science
Option: Software Engineering

Theme

Implementation of a heterogeneous data ingestion framework in a DATA LAKE environment

Prepared by
HADFI AMEL
AMOUBOUDI DYHIA

Supervised by
Professor: M.BALA

Defended on 03/10/2021,Committee composed of:

Ms Y.MANCER,MAA,Department of Informatics, Blida 1 President
Mr S.BENAISSI,MAA,Department of Informatics, Blida 1 Reviewer
Mr M.BALA,MCB,Department of Informatics, Blida 1 Promoter

Promotion: 2020/2021
Session: September 2021

Acknowledgement

We would like to thank first and foremost Allah the Almighty for His never-ending grace. Our deep gratitude goes to Professor Mahfoud Bala, who expertly guided us through our thesis project and shared his excitement and passion regarding the world of Big Data. His unwavering enthusiasm for this field kept us constantly engaged with our research and has indeed motivated us to continue our future career in this path.

Above ground, nobody has been more important to us in the pursuit of this project from than our families. We are indebted to them, whose value to us only grows with age. We would like to thank our parents, whose love and guidance are with us in whatever we pursue. Their support wasn't only financial but also emotional.

Abstract

The main purpose of this thesis paper deals with large and heterogenous formats of data. The reason behind why Big Data is so immense goes back to the five V's: Variety, Veracity, Volume, Velocity and Value. Our research aims to tackle the Variety and Value aspect of big data. Compromised within our research, we will be working in a Data Lake environment. DL's are made up with several components such as; Data Ingestion, Meta Data, Data Governance, Data security, etc. The module we have chosen to work on is Data Ingestion. Our study's aim is to ingest massive volumes of information from various sources into a Lake environment. To ingest our data, we will be implementing the Extract, Load, Transform (ELT) process instead of Extract, Transform, Load (ETL). The reason behind this decision was because we're working in a Data Lake environment, so data must be loaded in AS IS format with light transformations only. After exploring various data ingestion frameworks, we came across several solutions. The one that stood out from the crowd was Apache Spark. After thoroughly analyzing the framework, we found a couple of missing elements. After adopting Sparks framework, we proceeded to extend it by adding two of our features. The first is a Data Classifier and the second is a Data Visualizer. The new data ingestion platform has been developed in PyCharm IDE, Apache Spark 3.0.0, using Python 3.6, under Ubuntu 20 and the Data Lake we chose is Hadoop.

Keywords:
Data Lake, Data Ingestion, ELT, Data Classifier, Data Visualizer and Big Data.

Résumé

L'objectif principal de ce raport de thèse porte sur des formats de données volumineuses et hétérogènes. La raison pour laquelle le Big Data est si immense remonté aux cinq Vs: Variété, Véracité, Volume, Vélocité et Valeur. Notre recherche vise à aborder l'aspect variété et Valeur des mégas données. Compromis dans nos recherches, nous travaillerons dans un environnement data Lake. Les DL sont constitués de plusieurs composants tels que Ingestion de données, Métadonnées, Gouvernances des données, Sécurité des données, etc. La partie sur laquelle nous avons choisie de travailler est l'ingestion de données.
L'objectif de notre étude est d'ingérer des massifs volumes d'informations, provenant de diverses sources dans un environnement lacustre.

Pour ingérer nos données, nous mettrons en œuvre le processus Extract, Load, Transform (ELT) au lieu d'Extract, Transform, Load (ETL) à La raison que nous travaillons dans un environnement data Lake, les données doivent donc être chargées au format AS IS avec des transformations légères uniquement. Après avoir exploré divers frameworks d'ingestion de données, nous avons trouvé de plusieurs solutions. Apache Spark était à la tête de pertinence.

Après une analyse approfondie du cadre, nous avons trouvé quelques éléments manquants. En adoptant le framework Sparks, nous vons procédés à son extension en ajoutant deux de nos contributions. Le premier est un classificateur de données et le second est un visualiseur de données. La nouvelle plate-forme d'ingestion de données a été développée dans PyCharm IDE, Apache Spark 3.0.0, en utilisant Python 3.6, sous Ubuntu 20 et le data Lake que nous avons choisi est Hadoop.

Mots clés:
Data Lake, L'ingestion des données, ELT, Classifieur, Visualiseur, Données Massives.

الملخص

الهدف الرئيسي من هذه المقالة البحثية هو التعامل مع التنسيقات الكبيرة وغير المتجانسة من البيانات. يعود السبب وراء ضخامة البيانات إلى الميزات الخمسة للبيانات: التنوع ، الحقيقة ،الحجم ، السرعة و القيمة
يسعى بحثنا إلى معالجة جانب التنوع والقيمة للبيانات الضخمة. بعد بحوث سنتطرق للعمل في بيئة بحيرة البيانات. تتكون بحيرة البيانات من عدة أساسيات أهمها: استيعاب البيانات , البيانات الوصفية, إدارة البيانات وأمن البيانات وما إلى ذلك. الوحدة التي اخترنا العمل عليها هي "استيعاب البيانات".
تهدف دراستنا إلى استيعاب كميات هائلة من المعلومات من مصادر مختلفة في بيئة البحيرة.
لاستيعاب بياناتنا ، سنقوم بتنفيذ عملية الاستخراج والتحميل والتحو يل بدلاً من الاستخراج التحويل والتحميل, إحتراما لمبدئ العمل على بحيرة البيانات لذلك يجب تحميل البيانات بتنسيق
مع تحويلات جد قليلة و ضئيلة فقط (AS IS).
بعد استكشاف العديد من أطر استيعاب البيانات ، توصلنا إلى العديد من الحلول. االآداة التي تميزت و طغت عن البقية كان Apache Spark.
بعد تحليل شامل و اعتماد إطار العمل ، وجدنا نقائص. أردنا تمديده بإضافة ميزتين من ميزاتنا. الأول هو مصنف البيانات والثاني هو مصور البيانات. تم تطوير النظام الأساسي الجديد لاستيعاب البيانات في Python 3.6 ، Apache Spark 3.0.0 ، PyCharm IDE وبحيرة البيانات التي اخترنا مقرها في Hadoop تحت نظام التشغيل Ubuntu 20

الكلمات الدالة
.مصنف البيانات، مصور البيانات و البيانات الضخمة ،ELT ،بحيرة البيانات، ابتلاع البيانات

Contents

List of Figures

List of Tables

Abbreviations List

DL: Data Lake
DW: Data Warehouse
BD: Big Data
ELT: Extract, Load and Transform
ETL: Extract, Transform and Load
OLAP: OnLine Analytical Processing
OLTP: OnLine Transaction Processing
ROLAP: Relational Online Analytical Processing
MOLAP: Multidimensional Online Analytical Processing
CDC: Change Data Capture
BI: Business Intelligence
RDBMS: Relational Database Management System

General introduction

Context

In the period of data and information, the idea of big data is no longer something that fascinates businesses and society. For many years, big data has existed. It has been made clear that through big data solutions, establishments produce insights and make informed decisions, take notice of new trends and improve efficiency.

Big data can be defined as: “high-volume and variety of information assets that are generated at an extremely high speed”. The word ‘big data’ refers to data sets that are complex and enormous in terms of quantity. Market trends, unknown associations, patterns, customer preferences, etc. can all be found in big data, and need to be discovered. Occasionally, Big Data defines the process of gathering and examining great quantities of information, allowing businesses to make knowledgeable decisions. Big data cannot be handled by old and traditional data processing apps since it heterogenous.

Massive data can be created and collected from numerous sources, including social media networks, websites, mobile applications, text messages, geographical locations and other media files, etc. Sensors and Internet-of-Thing’s devices are the main drivers for the exponential growth of our data. If analyzed correctly, these data points can explain a lot about our behavior, personalities, and life events. The notion of big data and its importance has been around for years, but only lately has technology permitted the speed and efficacy at which immense sets of data can be analyzed. Both structured and unstructured grows significantly in the next few years, it will be collected and observed to expose unexpected insights and even benefit the prediction of our future.

The generation of large data at extremely short period of time has woken up a question: Where and how can we store it? The big data solutions framework is understanding big data storage methods on an enterprise level. This means being able to store and manage virtually unbounded volumes of data. Big data is often stored in a Data Lake. While data warehouses are commonly built on relational databases and contain structured data only, data lakes can support various data types and typically are based on Hadoop clusters, cloud object storage services, NoSQL databases or other big data platforms. The data in big data systems may be left in its raw

form and then filtered and organized as needed for particular analytics uses. In other cases, it's preprocessed using data mining tools and data preparation software so it's ready for applications that are run regularly.

Since data is so massive, heterogenous and is arriving at a fast speed, it has been quite difficult to load it into a data lake repository. The aim of data ingestion solutions was to aid scientists to make data ingestion somehow unchallenging. Even though the most difficult area when dealing with big data is finding how to ingest it since it's not a simple task to do and requires experts. The most popular data ingestion solutions that were created are: Apache Spark, Apache Gobblin, Marmaray and Lambda. These solutions have given a hand to experts to aid them in their difficult task to load data into the lake.

Even though all these solutions that were listed above do in fact effortlessly ingest data into the lake, they all share the same inconveniences in common. The cons that we found when studying these solutions are that they all ingest data in an extremely chaotic manner. These solutions also load data into the lake in an irrational method. The techniques and methods used in these generated solutions have led to un ultimate mess in the data lake leading it to lose rationality and value. The essential reason behind why data lakes turn in to swamps of data is a result of careless organization of data.

Problematic

In the last decade, we have witnessed a widespread use of Data Warehouses in particularly for decision making and data analysis. We already know it is without doubt that warehousing permits in loading preprocessed data to guarantee high performance when processing analysis on data. In addition, data warehousing is essentially storing data while using ETL process (Extract, Transform and Load). However, with the arrival of big data, the requirements for data analysis have become tremendously complicated and problematic to determine. Unquestionably the main issue with data warehouse that it's in continuous change thus making it unstable when it comes to making decisions. Also, data warehousing has found complications when loading various formats of data, since each format requires special treatment. In addition to the list of limitations, we often fail to estimate the time needed to retrieve, clean and load the data to the warehouse. ETL tools are here to make the process faster however efficient transformation takes up to several days or weeks.

To every problem there is a solution and that is Data Lakes. In a Lake environment, the structure of data can vary from structured, unstructured or semi-structured. The special feature of a Data Lake is that it loads data in its raw, untouched state also known as "As is" format. Since Data Lakes use ELT to process data, it requires less time to complete the overall procedure because we're loading the data directly without applying any transformation thus making it less time consuming. The ultimate path to solve this issue to create a framework that will load heterogenous data into a Data Lake through ELT tools.

Data ingestion is one of the most essential layers in a Lake's architecture. It's responsible to ingesting various formats of data that is arriving at different rates. We've decided to focus our research in this specific fragment of the Data Lakes architecture. We wish to implement Apache Sparks framework since its one of the most popular ingesting frameworks to exist and develop it to increase the overall ingestion operation.

Objectives and Contributions

As a part of our thesis project, we are keenly interested with working on massive and varied data in a Data Lake environment. Data Lake's are built up with many aspects such as, Data Ingestion, Governance, Storage, Security, Analytics etc. The component we have selected to work on and develop is Data Ingestion in a Lake environment.

The ingestion of data is effectively ingesting enormous sets of diverse data whether they're structured, unstructured or semi-structured data into a Data Lake. The procedure that will take place in our ingestion of data is ELT (Extract, Load Transform) and not ETL (Extract, Transform Load). Since we will be working in a Data Lake environment, we will be ingesting data in "AS IS" format.

The framework we wish to adopt and develop is Apache Spark. We will be appending our personalized features to improve the overall ingestion procedure in this framework. The two elements that will be added are Data Classifier before the data is ingested in to our Data Lake and Data Visualizer after the ingestion is completed. By integrating the data classifier/categorizer and visualizer to Apache Spark's solution, we will be solving the data organization issue that all the solutions share in common. We aim to solve this matter by operating the classifier to place our data in defined categories according to their format before the ingestion begins.

The data categorizer will also be ingesting and loading the data into defined categories in to the Lake. This means that we will gain total order on the Lake itself and reduce the risk of it transforming into an ocean of data. One of the biggest reasons for data lakes to fail is the lack of planning. Organizations dump all company-related data into their data lakes – this should not be done.

When placing the old approach next to our new approach, you can notice the difference vividly. With the old approaches, data was ingested in a random manner into a data pipeline and that was loaded directly into the data lake. Data that is ingested by these solutions and dumped directly into the lake means there is a lack of organization. This leads us to understanding there is a high risk of the Lake losing its value and transforming into an ocean of data. Secondly, Data Lakes becoming disorderly will have major influence on the user, since without doubt they will experience a tough challenge when searching for specific pieces of information or finding their precise location. Thirdly, without proper visualization of data and especially when it's in a chaotic state, this will lead to difficulties in understanding the meaning of data to a normal user.

In contrast with our new approach, categorizing the data into various data pipelines before ingesting it will help gain total control of the order in regards of the way data is getting ingested. Categorizing the data before ingestion has also aided in loading the data into structured and defined classes according to their format. This has also benefited the data lake in guarding its order and minimizing the chances of it turning into a swamp. By classifying the data in the lake, this will help users spend less time in understanding where their data is located therefore meaning they're saving time searching for information. In addition, to benefit the user in understanding

the data inside the lake, we've implemented the data visualizer after the ingestion is completed. This will benefit the end user in identifying patterns, better analysis, finding errors, grasping the latest trends and the most important aspect is understanding the story.

Motivation

One of the five key components of a Data Lake's architecture is Data Ingestion. The ingestion layer is a highly scalable layer in the system that extracts data from various sources. It should be flexible to run in batch, one-time, or real-time modes, and it should support all types of data along with new data sources. Throughout our thesis project we implemented the ingestion of heterogeneous data in a Data Lake environment. The reason why we decided to work in a Data Lake environment is because: Data lakes have become the foundation of many big data enterprises, just as they offer easier and more flexible options to scale when working with high volumes and varied data that's being generated at a high velocity – such as web, sensor or app activity data. Since these types of data sources have become progressively dominant, interest in data lakes has also been growing at a rapid pace Also, the reason behind why we selected data ingestion out of all the components of a Data Lake architecture is because many companies face major challenge coming from the massive generation of information from multiple data sources. Here is where data ingestion plays the main role by consolidating data and storing it in the Lake. Data ingestion is one of the primary stages of the data handling process. By using appropriate data ingestion tools companies can collect, import, process data for later use or storage in a Lake.

Thesis Organization

To effortlessly present our thesis work, our report has been precisely structured into eight chapters.

The first chapter possesses details regarding Data Warehouses. This chapter explains the function, types, schemas and architecture of DW. It also includes a detailed description of OLAP and OLTP operations.

The second chapter contains all the fundamental data such as: The concept of big data. We answered questions to what makes data tremendously big and what are its advantages. We went in depth and explained big data's ecosystem and all the components of its architecture, including ELT/ETL ingestion techniques.

Chapter three is where we break down the details of Data Lake environment. This chapter holds information such as: Importance, value, principals and the different technologies of a DL. To close off, in a table, we compared both Warehouses and Lakes.

The following chapter is Data Ingestion, here we described all the different layers to ingest data. In addition, Data Pipelines, CDC, data ingestion parameters and ingestion tools (Sqoop, Kafka, Fluentd & SAS) were all clarified in chapter five.

Compromised within chapter six are the various data ingestion solutions we found such as: Apache Spark, Gobblin, Marmaray and Lambda. To end this chapter, we provided a table to compare the solutions and select the one we found suitable for our requirements.

Chapter seven holds information concerning our contributions that were added to enhance the overall data ingestion procedure. Here, we've unraveled the two features that have been integrated with precise explanations and definitions. Finally, the last chapter is the implementation stage to finalize our theories. In this chapter, we present screenshots and detailed explanation of our application.

Finally, chapter eight holds the general conclusion of the entire report. Throughout this chapter we have managed to summarize every topic and subject that was explained. In addition, we've added a segment called Future Work. In this fragment, we have explained our work that will be integrated in the future.

PART 1

The Fundamentals

Chapter 1

Data Warehouse

Amid the first chapter, the information is all related to data warehouse. After a quick introduction to the topic, follows all the steps related to how the process of data warehousing functions. After that, all the types of data warehouses that are offered followed with their detailed definition. Next, the list of schemas that are implemented in the warehouse with their illustrations. Subsequently, we've presented the DW architecture and all the different procedures included (OLAP and OLTP). To close this chapter, we've laid down the advantages and disadvantages of warehouse and a conclusion.

1.1 Introduction

Data Lakes and Data Warehouses are both widely used for storing big data, but they are not interchangeable terms. A data lake is a vast pool of raw data, the purpose for which is not yet defined. A data warehouse is a repository for structured, filtered data that has already been processed for a specific purpose.

The two types of data storage are often confused, but are much more different than they are alike. In fact, the only real similarity between them is their high-level purpose of storing data. The distinction is important because they serve different purposes and require different sets of eyes to be properly optimized. While a data lake works for one company, a data warehouse will be a better fit for another.

The reason behind why we have a whole chapter dedicated to data warehouse is because we would like to clarify the difference between a warehouse and a lake. In addition, we aim to break down the different aspects of data warehouses, architecture, principals and the process they use to store data and accordingly compare them to a data lake.

A Data Warehouse (DW) is a procedure for collecting and handling data from diverse sources to provide meaningful business insights. A Data Warehouse is usually used to connect and analyze business data from heterogeneous sources. The data warehouse is the core of the BI system which is built for data analysis and reporting.[10]

An average data warehouse regularly contains the following elements:

- A relational database to store and manage data.
- An extraction, loading, and transformation (ETL) solution for preparing the data for analysis.
- Statistical analysis, reporting, and data mining capabilities.
- Client analysis tools for visualizing and presenting data to business users.

1.2 Data Warehouse Functionality

Information arrives from one or many data sources and is placed in a central repository. Data flows into a data warehouse from the transactional system and other relational databases. Data may be structured, semi-structured or unstructured data. Data Warehouse stores data in files or folders which helps to organize and use the data to take strategic decisions.[2]
The important functions which are needed to perform are:

- Data Extraction
- Data Cleansing
- Data Transformation
- Data Loading and Refreshing

The data is processed, transformed, and ingested so that users can access and work with the processed data in the Data Warehouse through the help of many tools. A data warehouse combines information coming from different sources into one comprehensive database. By merging all of this information in one place, this aids organization to analyze its customers more accurately. Data warehousing also makes data mining possible.

1.3 Categories of Data Warehouse

1. **Enterprise Data Warehouse (EDW):** Is a centralized warehouse.It offers decision support service across the enterprise, an integrated approach for organizing and representing data. It also provides the capability to classify data according to the subject.[13]
2. **Operational Data Store (ODS):** In ODS, Data warehouse is refreshed in real time. Hence, it is widely preferred for routine activities like storing records of the Employees.
3. **Data Mart:** A data mart is a subset of the data warehouse. It is specifically designed for a precise line of business, such as sales or finance. In an independent data mart, data can collect directly from sources.

1.4 Data Warehouse Schemas

The Data Warehouse Schema is a structure that reasonably defines the contents of the Data Warehouse , by facilitating the operations performed on the Warehouse. A schema is a logical description that describes the entire database.[13]
There are three types:

1. **Star schema** It is a really basic structure to store the data in the data warehouse. The center of this start schema one Fact table which links a series of Dimension tables. It is crucial to fully understand the fact and dimension tables to be able to comprehend the star schema.[2]
   An example of a star schema is illustrated in Figure 1.1

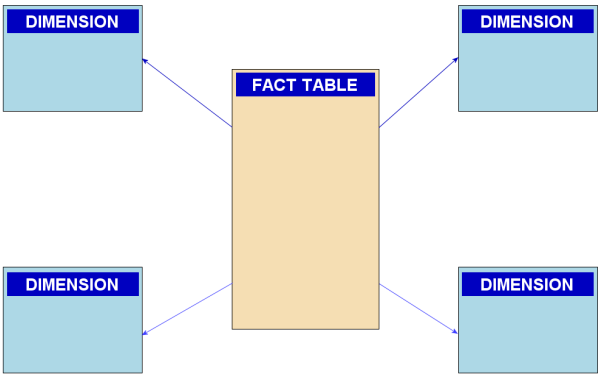

Figure 1.1: Star-schema-illustration [27]

2. **Snowflake schema** The snowflake schema is the multidimensional structure. Similar as the star schema the Fact table connects to the Dimension table but the only difference to achieve the snowflake schema is to divide the Dimension tables into sub-dimension tables. It is a very complex database design but it manages to keep a low-level data redundancy. An example of a snowflake schema is described in Figure 1.2 below.

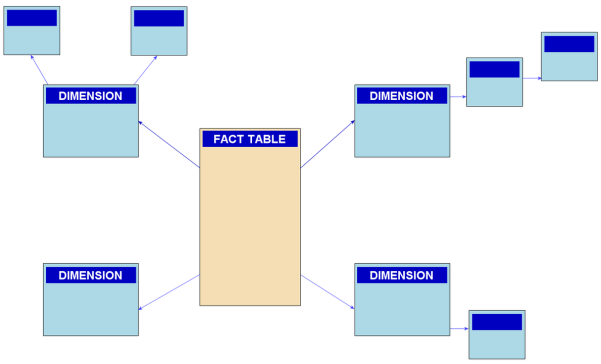

Figure 1.2: Snowflake-schema-illustration [27]

3. **Galaxy/constellation schema** The constellation schema means two or more Fact tables sharing one or more Dimensions. This schema describes a logical structure of data warehouse or data mart. It is a sophisticated database design that is difficult to summarize information.
   Figure 1.3 is a representation of a galaxy schema.

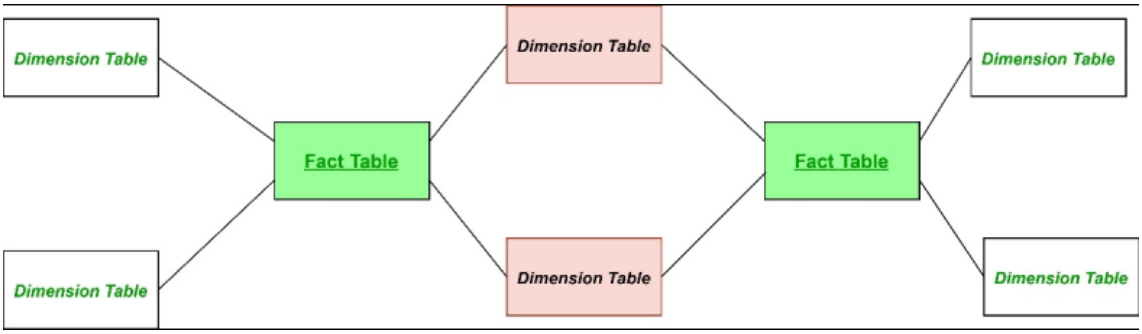

Figure 1.3: Galaxy-schema-illustration [28]

1.5 Data Warehouse Architecture

There are three approaches for constructing Data Warehouse layers: Single Tier, Two tier and Three tier.

- **Single-tier architecture:**The main objective of this architecture is to reduce the amount of data stored and remove data redundancy. This architecture is not frequently used.
- **Two-tier architecture:** This type of architecture separates physically accessible sources and data warehouse. This architecture faces a lot of issues which are: It is not expandable, does not support a large number of end-users and has connectivity problems because of network limitations.
- **Three-Tier Architecture:** This is the most widely used Architecture of Data Warehouse. It consists of the Top, Middle and Bottom Tier.

  - **Bottom Tier:** Is the database of the warehouse. It is usually a relational database system. The data is cleansed, transformed, and loaded into this layer.

  - **Middle Tier:** The middle tier in Data warehouse is an OLAP server which is implemented using either ROLAP or MOLAP model.

  - **Top-Tier:** The top tier is a front-end client layer. This tier contains all the tools and API that you connect and get data out from the data warehouse. It could

be Query tools, reporting tools, managed query tools, Analysis tools and Data mining tools.

1.6 OLAP(Online Analytical Processing)

OLAP is a software that lets users to analyze information from several database systems at the same time. It is a technology that allows analysts to extract and view data from different points of view.

OLAP Cube: Figure 1.4 below, is an illustration that describes an abstract version of the OLAP cube. It is a data structure which has been adjusted for extremely quick data analysis. A Data warehouse would extract information from various data sources. After that, the extracted data is cleaned and transformed. Later on, data is loaded into an OLAP server or OLAP cube where information is pre-calculated in advance for further analysis.

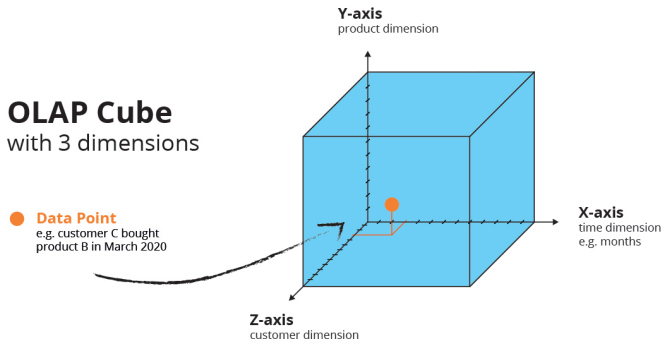

Figure 1.4: OLAP-cube-description [29]

1.6.1 OLAP Operations

Roll-up

Roll-up is also known as "aggregation". The rollup operation performs aggregation on a data cube. It can be performed in two ways

- Reducing dimensions.
- Climbing up the hierarchy. Concept hierarchy is grouping things based on their order or level.

Drill-down

It is data that is broken down into smaller chunks. It is the opposite of the rollup process. It can be done via increasing dimension or moving down the concept hierarchy

Slice/Dice

To slice and dice is to break a part of the information down into smaller parts or to study it from different viewpoints so that you can understand it better.

Pivot

Pivot otherwise known as Rotate, changes the dimensional orientation of the cube; it rotates the data axes to view the data from different perspectives. Pivot groups data with different dimensions.

1.6.2 Advantages and Disadvantages of OLAP

Table 1.1 underneath presents two columns. The first column presents the advantages and the second column presents the disadvantages that come with OLAP processing,

Advantages	Disadvantages
• OLAP includes planning, budgeting, reporting, and analysis. • Information and calculations are consistent in an OLAP cube • Quickly create and analyze "What if" scenarios. • OLAP provides tools such as modeling, Data mining and performance reporting forbusiness • It is a powerful visualization online analytical process system which provides fasterresponse times.	• OLAP requires data to be organized in either a star or snowflake schema. Theseschemas are difficult to implement. • In an OLAP cube you cannot have a large number of dimensions. • OLAP system cannot access transactional data. • The cube must be updated after any modification. This is a time-consuming process.

Table 1.1: Advantages and Disadvantages of OLAP

1.7 OLTP (Online Transaction Processing)

OLTP is an operational system that supports transaction-oriented applications in a three-tier architecture[13]. It manages the everyday transactions of an organization. It enables the real-time execution of large numbers of database transactions by large numbers of people, typically over the internet. OLTP has placed the spotlight on query processing, preserving data integrity and the effectiveness that is measured by the total number of transactions per second. Below is a list of OLTP characteristics:

- OLTP uses transactions that include small amounts of data.
- Process a large number of relatively simple transactions:insertions, updates, and deletions to data, and simple data queries.
- The response time of OLTP system is short and very rapid.
- End-users can access directly to databases.
- OLTP uses a fully normalized schema for database consistency.
- It supports complex data models and tables.

1.8 Advantages and Disadvantages of Data Warehouse

Table 1.2 below represents the pros and cons of a Data warehouse.

Advantages	**Disadvantages**
• Data Warehouse aids to integrate many sources of data. • Data warehouse allows users to access critical data from the number of sources in a single place. Therefore, it saves user's time of recovering data from several sources. • Data warehouse stores a large amount of historical data. This helps users to analyze different time periods and trends to make future predictions.	• Not an ideal option for unstructured data. • Creation and implementation of Data Warehouse is time consuming. • Data Warehouse can be outdated relatively quickly. • Difficult to make changes in data types and ranges, data source schema, indexes, and queries. • The data warehouse may seem easy, but actually, it is too complex for the average users.

Table 1.2: Advantages and Disadvantages of Data Warehouse

1.9 Conclusion

Data warehousing is the foundation of automatic decision support system. It has been examined a lot in the past ten years but still there are countless problems to be tackled in future. Performance and organization are between the top research questions now. We have recognized some of the newest tools available for data warehousing and classified the tools in rational manner. The architecture of the data warehouse is also divided logically as well as a typical model of the architecture is also given. We further analyzed some of the major research areas like OLAP and OLTP. Finally, in a table placed a couple of benefits and inconveniences that concerns data warehouses.

Chapter 2

Big Data concepts

In this chapter, a thorough explanation of the basic concepts of big data. Also, the five V's that define why data is so big are listed below with detailed explanation. During this chapter we will be presenting the challenges of Big Data and the advantages that come with it. In addition, a detailed description on the different types of data (Structured, unstructured and semi-structured). Correspondingly, this chapter contains a complete and detailed explanation on big data's ecosystem and all the different components that are held inside of it. This chapter ends with a conclusion to sum up everything we discussed

2.1 Introduction

Big data is a term that describes the large volume of data – both structured and unstructured – that engulfs a business on a day-to-day basis. But it's not the amount of data that's vital. It's what organizations do with the data that matters. Big data can be analyzed for insights that lead to better decisions and strategic business moves.

Big Data helps the organizations to create new growth opportunities and entirely new categories of companies that can combine and analyze industry data. These companies have plenty of information about the products and services, buyers and suppliers, consumer preferences that can be captured and analyzed.

The term "big data" refers to data that is so large, fast or complex that it's difficult or impossible to process using traditional methods. The volume, velocity and variety of big data is what makes it so "Big".

2.2 V's in Big Data

Big Data is broken down into five categories known as 5 V's; Volume, Velocity, Variety, Veracity and Value as portrayed in Figure 2.1

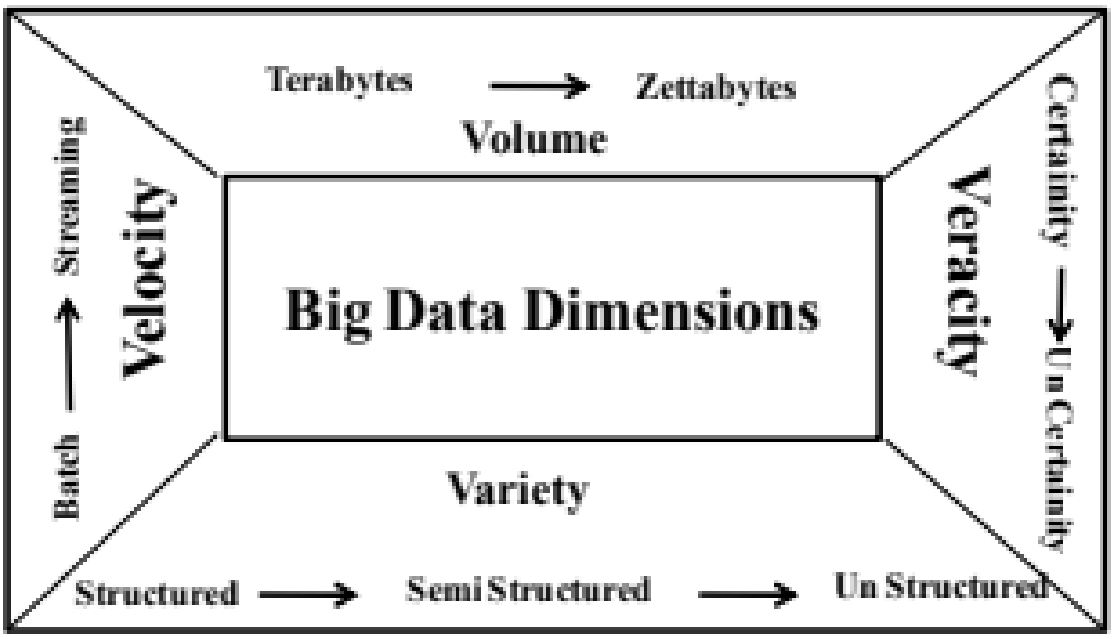

Figure 2.1: V's in Big Data[14]

Volume

Volume refers to the incredible amounts of data generated each second from social media, cell phones,cars, credit cards, photographs, video, etc. There's a lot of data out there and it comes in large amounts of data sizes of terabytes to zettabyte which is an almost incomprehensible amount. With 90 Value When we talk about value, we're referring to the worth of the data being extracted.[17].

Value

When we talk about value, we're referring to the worth of the data being extracted. The majority of Data having no Value is of no good to the company, unless you turn it into something useful. Having endless amounts of data is one thing, but unless it can be turned into value it is useless.

Valocity

Velocity refers to the speed at which vast amounts of data are being generated, collected and analyzed. It is the speed of which the data arrives ready to be processed. Every day the number of emails, twitter messages, photos, video clips, etc. increases at lighting speeds around the world. Every second of every day data is increasing. Not only must

it be analyzed, but the speed of transmission, and access to the data must also remain instantaneous.

Variety
Data is big, data is fast, but data is also extremely diverse. Data today looks very different than data from the past. We no longer just have structured data (name, phone number, address, financials, etc.) that fits nice and neatly into a data table. Today's data is unstructured. In fact, 80% of all the world's data fits into this category, including photos, video sequences, social media updates, etc.

Veracity
Veracity refers to the accuracy of data. Not all data is accurate or dependable, and with the development of big data, it's becoming harder to control which data brings which value. »

2.3 Big Data Challenges

As promising big data sounds, it comes with its challenges. First of all, it is called big data for a reason. Although new technologies have been developed for data storage, data volumes are still doubling in size. Big companies are still struggling to keep pace with their data and are continuously finding ways to effectively store it.But it's not enough to just store the data. Data must be used to be valuable. It requires a lot of analysis and work to bring meaning to data. This includes the cleansing of data and finding relevance within it. Data scientist say that they would spend 50% to 80% of their time preparing data before it can actually be used. Finally, the technology that is being used for big data is changing at tremendous speed. A few years ago, the most popular technology that was used to handle big data was Apache Hadoop. After that in 2014 Apache Spark was introduced. Today, the best approach when dealing with big data would be the combination of the two frameworks to keep up with the challenges faced.

2.4 Benefits of Big Data

- Big data makes it possible for you to gain more complete answers because you have more information.
- More complete answers lead to more confidence in the data which means a completely different approach to tackling problems.
- Identification of important information that can improve the quality of decision making.
- It can provide ideas from huge amounts of data from multiple sources that include those that come from external third-party sources, the internet, social networks, those already stored in company databases etc.

- Big data is helpful in keeping data safe.

2.5 Types of Big Data

Structured data

Structured data is the only language a computer is capable of understanding. It concerns all data which can be stored in database SQL in table with rows and columns. Structured data represents only 5 to 10% of all informatics data. Structured data is the easiest type of data to analyze because it requires little to no preparation before processing. The ETL process for structured data stores the finished product in what is called a data warehouse[8]. These databases are highly structured and filtered for the specific analytics purpose the initial data was harvested for.

Unstructured data

Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data. Unstructured data represents around 80% of data. It often includes text and multimedia content. Examples include e-mail messages, word processing documents, videos, photos, audio 10files, presentations, web pages and many other kinds of business documents. Just as with structured data, unstructured data is either machine generated or human generated. The hardest part of analyzing unstructured data is teaching an application to understand the information it's extracting. More often than not, this means translating it into some form of structured data. Products like Hadoop are built with extensive networks of data clusters and servers, allowing all of the data to be stored and analyzed on a big data scale.

Semi-structured data

Semi structured is the third type of big data. Semi-structured data pertains to the data containing both the formats mentioned above, that is, structured and unstructured data. Semi-structured data is information that doesn't exist in a relational database but does have some structural properties that make it easier to analyze. With some process you can store them in relation database. For example, XML files or emails are examples of semi-structured data. Semi-structured data has no set schema. This can be both a benefit and a challenge. It can be more difficult to work with because effort must be put in to tell the application what each data point means. But this also means that the limits in structured data ETL in terms of definition don't exist.

2.6 Big Data Ecosystem

It's not simple to extract data and it automatically turns valuable, it goes through many processes of analysis that turn it into insights. Big data components pile up in layers, building a stack. It's a long difficult process that can take up to months maybe even years to implement. Data must first be ingested from heterogeneous sources, translated and stored, then analyzed before final presentation in an understandable format. Figure 2.2 represents the essential fragments inside a big data ecosystem. Taking a look at the image below, the first two layers of a big data ecosystem, ingestion and storage, include ETL. Extract, transform and load (ETL) is the process of preparing data for analysis. Concepts like data extraction, loading, transforming all describe the pre-analysis prep work. Working with big data requires significantly more prep work than smaller forms of analytics.

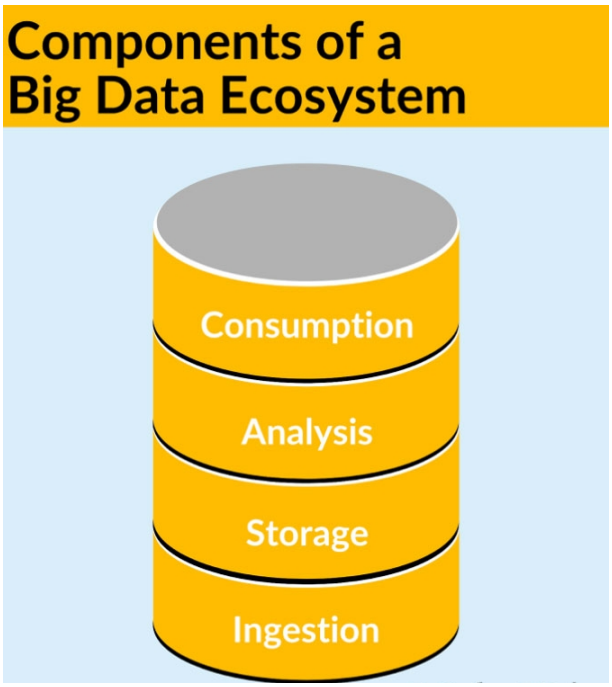

Figure 2.2: Components-of-Big-Data Ecosystem [30]

2.6.1 Ingestion

The first step is dragging in data in its raw form, is the ingestion layer. Data can come from relational databases, social media, emails, phone calls or somewhere else. There are two kinds of data ingestion:

Batch, in which data is gathered in large bulks and delivered together at the same time. The collection of data can be triggered by conditions launched on a schedule.

Streaming, it is data that arrives in real time which is extremely necessary for data analytics. As data is being generated it is also being pulled for direct analysis. Streaming data requires more resources to constantly monitor any changes in data pools.

It's not all about just getting the data into the system. This presents lots of challenges, some of which are:

- Sustaining security: With a lot of data flowing in, it is no question in mind that it is a challenge to assure that every single dataset doesn't carry some sort of security vulnerabilities. All data must be retrieved ethically and according to the law, which can be problematic to manage with such great quantities.
- Guaranteeing data quality: Just because there is a hefty amount of data doesn't mean it's all valuable and useful. Having too much irrelevant, incomplete or incorrect data can cause issues in analysis and processing down the line.
- Data speeds:Each data source has different setup for transporting data. If a data source is slow and comes with low resources in exporting, this can drag down the entire speed of the process and introduce new errors.

2.6.2 Data Cleansing and Organization

When data arrives, before it can be used for analysis it needs to be sorted and translated correctly. It is essential that we manage to get as close as possible to achieve a uniform organization to be able to process this data according to its timely manner in the actual analysis stage. The components in the storage layer are responsible for making data readable, homogeneous and efficient because data arrives in different formats and schemas. Once all the data is converted into readable formats, it needs to be organized into a uniform schema. A schema basically defines the characteristics of a dataset. For structured data, aligning schemas is all that is required. However, for unstructured and semi structured data, semantics are required to be given to it before it can be accurately organized. After all the data is as similar and close as can be, it then needs to be cleansed. This means eliminating all useless and irrelevant information within the data. When data comes from external sources, it's enormously common for some of those sources to duplicate each other. Therefore, useless information in the database must be removed from the dataset that will be used for analysis. Once all the data is transformed, organized and cleaned, it is ready for storage and analysis.

2.6.3 Storage

The last step of ETL is loading the converted data and storing it in either a data lake or warehouse and eventually processing it. It's the actual embodiment of big data: a huge set of usable, homogeneous data, as opposed to simply a large collection of random, incohesive data. It is considered that data lakes and date warehouse are the most crucial element of a big data ecosystem because they need to contain only relevant data and hold little redundancy to allow quicker processing. Lakes differ from warehouses since they preserve raw data which means little or no transformation has been applied. On the other hand, warehouses are focused on the specific task for other analysis efforts thus makes it store less data and produce quicker results. This also means that data lakes require a lot more storage and significant amount of transformation efforts down the line. This leads on to the modification of extract, transform and load: extract load and transform.[5]

2.6.4 ELT (Extract, Load and Transform)

When creating a data lake there is a process called Extract, Load and Transform (ELT) that is used. The data is not transformed until the analysis stage. By doing this, the initial integrity of the data is being preserved, meaning no information is getting lost in the transformation stage permanently. For this reason, data lakes are preferred for fetching unused data later on completing different queries and analysis on the complete dataset. Whereas with a warehouse, you most likely can't come back to the stored data to run a different analysis. ELT process: it is related to ETL (Extract, Load and Transform). Both of these processes involve three steps, but with a crucial an obvious difference in the order:

- **Extract:** Retrieve data from various and dispatched sources, such as databases, social media, cloud services, and other data repositories.
- **Load:** Move the data to a destination repository, such as a data warehouse (for structured data) or a data lake (for unstructured data)
- **Transform:** To improve quality of the data changes are applied to the data so that it can return accurate search results. Typical transformations are similar like ETL transformations (removing duplicates, irrelevant, missing data)

2.6.5 ETL (Extract, Transform, Load)

Of course, the ETL process is much more involved than the brief definition above. To build an ETL process, you need to answer questions and concerns about each of the three ETL stages, such as:

- **Extract:** Two questions need to be answered in this phase: which data will you extract and which kind of data sources? ETL data sources derive from numerous sources such as

relational and non-relational databases, computer files, websites, social media and more. However, not all data that is extracted means its valuable and relevant for your purpose. In this case, you'll need to decide which data to filter out, and organize it in a manner to push it through the ETL pipeline.

- **Transform**: Several changes and transformations occur in this phase. The list below presents a couple types of possible transformation of data:

  - *Cleansing:*eliminating old, irrelevant observations, handle missing data, and duplicated data.

  - *Joining:* merging data from several sources.

  - *Validating:* ensuring the integrity of the data.

  - *Summarizing:* performing calculations on existing data and then creating new data records.

- **Load:**In this phase, the destination of the final result of data should be define to where it can be stored. In most cases, ETL prefers to use data warehouse to stores the transformed data in a structured format plainly created for reporting and analytics. On the other hand, some ETL architectures use an unstructured data lake as their endpoint. The data in a data lake remains untouched and in its original format.

The main difference between ETL and ELT is that in ETL data is transformed before loading it. this means that the data that is sent to the data warehouse is neat and standardized data. However, in ELT the data is replicated without disturbing its natural format (no transformation performed). This means that the data in a data lake is an exact replica of the original data source. In ELT approach, transformation happens when a person or process needs something from the data lake. This strategy is schema-on-read which is the complete opposite of ETL's schema-on-write.

ETL VS ELT

Figure 2.3 is an illustration that compares the two different procedures ETL/ELT when loading data into a warehouse or lake. This illustration will help compare both in the bullet points underneath.

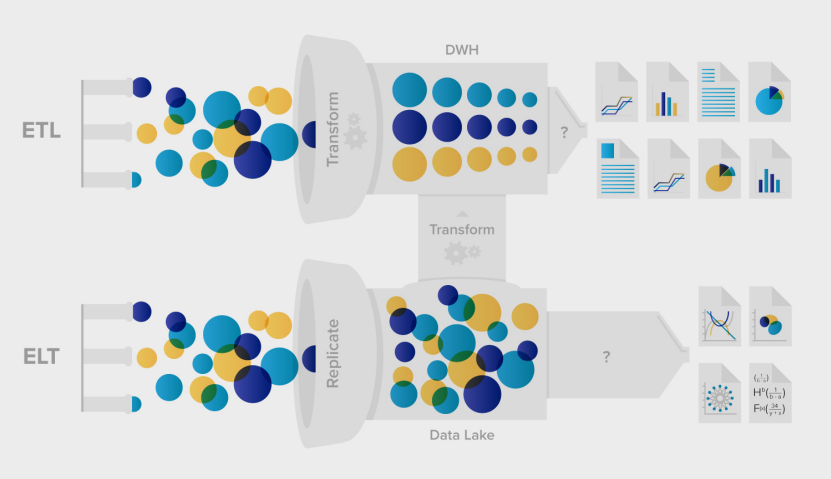

Figure 2.3: ETL vs ELT [21]

- ETL loads data first into the staging server and then into the target system whereas ELT loads data directly into the target system.
- ETL model is used for relational and structured data while ELT is used for scalable structured and unstructured data sources.
- ETL is mainly used for a small amount of data whereas ELT is used for large amounts of data.
- ETL is easy to implement whereas ELT requires high skills to implement and maintain.

2.6.6 Data Analysis

Data analysis is consisted of a couple of processes that come in the following order: collecting, analyzing, interpretating and visualizing data in a manner to discover and extract important insights to help conclude effective and smarter business decisions. All the hefty work happens in this big data component called Analysis.

There are several methods and techniques to accomplish the analysis and also taking in consideration of the industry and the aim of the analysis. The main purpose of data analysis is to find meaning in data. The process of data analysis uses analytical and logical reasoning to gain information from the data.

A basic example of Data analysis that we use in our day-to-day life without even realizing it would be, when it comes to taking any decision, we usually find ourselves thinking about our

past experiences and how the results turned out, we also try to predict what would happen in the future if we made a certain decision. For that, we gather memories, past experience or our thoughts of the future. All of this is nothing but analyzing our past or future and based upon that we make a decision. Basically, that is same thing analyst does for business purposes, it is called Data Analysis. After collecting, ingesting and preparing raw data, the next step it to pass this data through several tools to shape it into actionable insights.

2.6.6.1 Techniques and Methods of Data Analysis

Below is list of the major and popular types of analysis on big data:text, statistical, descriptive, diagnostic, predictive and prescriptive.

- **Text Analysis:**Text Analysis or Data Mining both mean the same thing. Data analysis has allowed us to discover patterns and trends throughout large sets of data throughout the usage of data mining tools or databases. In general, it provides a way to extract and examine the data, later on deriving patterns and trends and finally interpret the data.

- **Statistical Analysis:**This type of analysis allows us to answer "what happened?" questions by using data from the past in the form of dashboards. Statistical Analysis comes with the collection, analysis, interpretation, presentation, and modeling of data. There are two categories inside statistical analysis: Descriptive and inferential analysis.

- **Diagnostic Analysis:**Clarifies why a problem is happening. Big data allows analytics to take a deep dive into certain information's and indicators to explain why specific actions didn't give out the anticipated results.

- **Predictive Analysis:**Based on historical data, highlighting patterns and evaluating routes of relevant metrics it can predict and estimate future efforts and making it more reliable.

- **Prescriptive Analysis:**Takes predictive analytics to a whole new other level by highlighting the best future efforts. Prescriptive analytics allows businesses to decide how to put their best foot forward and that's by modifying inputs and changing actions.Different actions will give off different results, and prescriptive analytics helps decision makers to implement the best procedures.

Just as the ETL layer is evolving, so is the analysis layer. We can now discover insights impossible to reach by human analysis.

2.6.6.2 Data Analysis Tools

With the aid of data analysis tools, users have the ability to process and control data, discover patterns and trends and analyze the associations between data sets. Below is a list of tools used for data analysis.

Apache Spark

It is one of the most powerful, influential and most used open-source big data analytics tools. It has become easy to build parallel apps since it offers over 80 high level operators.

Features:

- Ability to Integrate with Hadoop and Existing Hadoop Data.
- It is one of the open-source big data analytics tools that provides built-in APIs in Java, Scala, or Python. Spark combines libraries including SQL, Data-Frames, MiLB for machine learning and spark streaming.
- It runs on Hadoop, Apache Mesos, Standalone or in the cloud.

R-programming

It is a language that provides a wide variety of statistical test, it that can be used for both computing and graphics plus big data analysis.

Features:

- Effective data handling and storage facility.
- Provides a list of operators for calculations on arrays and matrices.
- It offers graphical services for data analysis which display either on-screen or on hardcopy.

Talend

It is a big data analytics software that simplifies, automates big data integration, master management and also check the quality of data.

Features:

- Extremely quick to extract value for big data projects.
- Simplify ETL and ELT for big data.
- The usage of MapReduce and Spark is simplified by generating native code.
- Smarter data quality with machine learning and natural language processing.

2.6.7 Consumption

The final big data component includes presenting the information in a readable and understandable format to the end-user in this case it is the executives and decision-makers. This can appear in forms of tables, advanced visualizations and even single numbers if requested. Visualizations come in the form of real-time dashboards, charts, graphs and so on.

2.7 Conclusion

The chapter contains an overview of current big data technologies as well as its ecosystem. The overview specifically included the importance of big data and the challenges that tag along with it. Rather than presenting a broad overview of the ecosystems, we focused on the detailed descriptions of individual technologies that were provided. ELT/ETL approaches were also presented and explained completely.

It can be concluded that data passes through multiple check points during its life cycle. Since the beginning of it getting extracted from the source, passing through the ingestion pipeline, then applying some alterations through ELT and data analysis and being directed to the consumption layer to be manipulated again. In order to greet massive data with many formats and at great speed, the big data world is always on the hunt for new technologies to support its incoming data.

It can also be concluded that there is a strong need to increase the maturity of storage technologies so that they fulfil future requirements and lead to a wider adoption. Technologies similar to Hadoop have had an enormous impact on big data. Since HDFS has the capability to store all types of data (Structured, Unstructured & Semi-structured) in huge chunks and provide full security, many companies have chosen it as a storage unit for their data. In addition, tools are were presented in this chapter concerning data analysis have also changed the way data is presented. Like said, data without value isn't of much use to us, that's why applying data analysis on it and extracting hidden tools can help any company making the correct decisions and developing the overall outcome.

With the birth of new formats of data, it is not possible to mine and process this mountain of data with traditional tools, so we use big data pipelines to help us ingest, process, analyze, and visualize these tremendous amounts of data.

Chapter 3

Data Lakes

Presented in this chapter is a quick introduction to swiftly become integrated into this matter. We have discussed the importance and the characteristics of Data Lakes. We have answered questions: How to get value from a DL? What are the principals of DL? What technologies are presented in a DL? Finally, we have compared Data Lakes to Data Warehouses and ended the chapter with a conclusion to summarize everything included in this chapter.

3.1 Introduction

A Data Lake is a place to store and hold enormous volumes of raw data in its natural format until it is needed[9]. It is a centralized repository that allows you to store all your structured and unstructured data at any measure. You can store your data as-is, without having to first structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions. While a hierarchical data warehouse stores data in files or folders, a data lake uses a flat architecture to store data. Each data element in a lake is given a unique identifier or address and tagged with a set of extended metadata tags.
A few specific properties of a data lake:

- All data is loaded from source systems. No data is turned away.
- Data is stored at the leaf level in an untransformed or nearly untransformed state.
- Data is transformed and schema is applied to fulfill the needs of analysis.

3.2 Importance of Data Lake

Following up recent research, it has been concluded that businesses are seeing volume growth of their data that surpasses 50 % per year. Additionally, these businesses are juggling an average of 33 different data sources which are used for analysis. As data volumes, varieties, and velocities increase, the ability to firmly store, process and manage that data becomes more problematic over time. A data lake architecture has proven its values and characteristics allowing organizations to overcome these challenges by offering a centralized repository that lets the storage of business data no matter the volume, variety, or velocity at which it is created. Many analytical workloads are being served by this single repository such as visualizations and dashboards, machine learning and beyond.

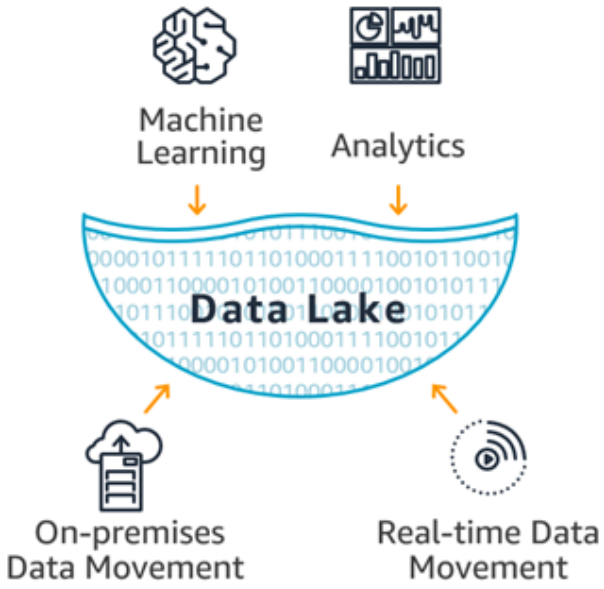

Figure 3.1: Illustration of Data Lake components[16]

3.3 Characteristics of a Data Lake

Figure 3.1 above is a representation of the fundamental Data Lake characteristics. Underneath is a list that includes the important features of a lake.

- It provides diverse analytics capabilities, including batch processing, stream computing, interactive analytics, and machine learning, along with job scheduling and management capabilities.[11]
- The main principal of a data lake is the centralization of its data. The benefits that come from centralization would include it being easier to govern, manage and innovate data sets.
- It can store massive amounts of data of all types, including structured, semi-structured, and unstructured data.

- A data lake delivers sufficient data storage to store all of the data of an enterprise or organization.
- A DL provides full metadata to manage all types of data-related elements, including data sources, data formats, connection information, data schemas etc.[18]
- A data lake provides full capabilities for data retrieval and publishing. It supports a wide variety of data sources.
- A data lake provides big data capabilities, including the great storage space and scalability needed to process data on a large scale.

3.4 Extracting Value from a Data Lake

The main ability of a DL is to store data of unknown value, for almost small cost. This historical data can be used to train machine learning models and answer questions in the future.
Nevertheless, for an organization to extract the most value out of a data lake is to use it as an engine for innovation. By making data access simpler, faster and more efficient for users and simplifying testing with different processing technologies, businesses can discover new insights that fire up competitive gain.

3.5 Principles of Data Lake Storage

High scalability
A data lake's main aim is to store centralized data for an entire company hence it must be capable of weighty scaling without running into limitation issues.

High durability
The prime repository of enterprise data, an extremely high durability of the storage layer permits for brilliant data strength without needing to pay for extreme high-availability designs.

Structured, unstructured and semi-structured data
A data lakes main design consideration is the capability to store data of all types in a single repository. Example XML, Text, JSON, CSV etc.

Independence from fixed schema
Applying schema on read as many times needed for each consumption objective is only accomplished if the core storage does not demand a fixed schema. Schema development is common in the big data era.

Cost Effective

Systems now have the ability to quickly scale as data grows because open source is basically free. Handling numerous formats of data and data models, parallel with suitable compression methods, are beneficial to avoid cost evolution exponentially. AWS, Google and Azure all offer object storage technologies. e.g., S3, Blob storage, ADLS etc.

3.6 Data Lake Technology

Data lake technology is being developed by a growing number of vendors who invest in Hadoop-related components, Google, Cloud Platform, Amazon Cloud and Microsoft Azure.[18] Taking a look at Figure 3.2 in this image are the common data lake technologies are represented: Hadoop, Spark, Hive, etc.

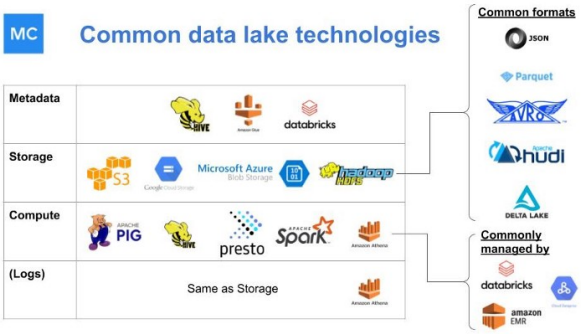

Figure 3.2: Data lake technology [31]

3.6.1 Storage

The basis of any Data Lake implementation is physical storage. The fundamental storage layer is used for the primary data assets which typically holds raw and/or lightly processed data. Processing data without taking in consideration it's form, size and quantity while its coming in streaming or in batches comes with its challenges concerning the storage infrastructure. The Hadoop Distributed File System (HDFS) has become popular for on-premises Data Lakes because it is reasonably low-priced and appropriate for batch processing workloads by Hadoop tools like MapReduce.

3.6.2 Data Sources

When it comes to selecting a data source for a Data Lake it is extremely important to know where data is coming from to help enrich analytical insights on a business statement. This is a couple of different data sources that can data be ingested into a Data Lake:

- Web content – social media platforms similar to Facebook, Twitter, Snapchat and Linked-In accumulate enormous amounts of data. This sort of data varies from structured to

un-structured data such as texts, images or videos that is used to study user's performance, business profiles, content and campaigns.

- Geographical details – data streaming from location data such as maps and geo-positioning systems (GPS).
- OLAP systems and relations data – stores structured data from relational database and it can be ingested directly into a data lake.
- Data management systems – semi-structured data such as documents and text files that are associated with business entity. These can be manipulated to fit in a structured format.

3.6.3 Data Ingestion

Data ingestion framework is about moving data and especially the unstructured data. This framework works by capturing data from numerous data sources and ingests it into a Data Lake. The data ingestion framework keeps the Data Lake consistent with the data changes at the source systems, making it a single station of enterprise data.[26]

3.6.4 Data Profiling Integration

The value of our data is determined on how well we profile it. Since data is stored in its original format, we have faced a data ingestion problem another issue would be that poorly managed data is costing companies a lot of time and money going to waste. Before a data scientist can extract results and begin the analysis phase, data must be profiled to be able to understand the structure, quality and the schema of the data itself. The following step that would be applied is schema-on-read or schema-on-write depending on the data format.

- Schema-on-read: We upload the data as it arrives without applying any transformations. This schema has fast data ingestion since it doesn't follow any internal schema. This means that it's just copying and moving files. This schema is more flexible and dynamic when it comes to big, unstructured data or schemas that are frequently changing. would be from the raw data we can extract important data then later on transform it in a way to allow correlation with other data.
  As fun as schema-on-read sounds it has its cons. Since the data doesn't go through strict ETL transformations it is vulnerable to missing or invalid data and having duplicates.
- Schema-on-write: This schema is tightly bound to relational database, this includes schemas, table creation and data ingestion. This means that data cannot be uploaded unless the tables and schemas have been created and configured. Additionally, the most time-consuming task when working on relational database would be the ETL work. A framework such as Hadoop does not require schema-on-write for unstructured data.

3.6.5 Processing

The data extraction process is accomplished by implementing the following stages that are completed by a data scientist

1. The data must be prepared to avoid any challenges that come with ingestion it.
2. Data analytics or machine learning have to applied.
3. Establishing results for consumption layer.

Frameworks that allow to parallelize and scale processing jobs are Hadoop MapReduce paradigm, Apache-Spark that allow batch or stream-based processing. In general, MapReduce based solutions are useful for batch processing and for analytics that are not real time or near real-time. A couple of operations a data scientist might consider executing when it comes to the three steps that were mentioned about would be: data profiling, filtering, data cleaning,data mining and so on.

3.6.6 Data Governance

Dumping data into a Hadoop platform won't automatically speed up your analytical efforts. Data Lakes if not monitored they have high risks of them turning in to "data swaps" The lack of control on data may put in risk data pipelines, quality, structure, sources, and the results of data. A governed Data Lake means that it holds clean, relevant data, structured and unstructured sources than can effortlessly be found, accessed, managed and secured. Full protection and reliability are the two main features your data must be offered by the platform it exists on. So,data that flows in to the lake must be cleaned, classified and secured.

3.7 Data Lake VS Data Warehouse

- Data Lake stores all data not taking in consideration of the source and its structure whereas Data Warehouse stores data in measurable metrics with their attributes.
- Data Lake outlines the schema after data is stored however Data Warehouse defines the schema before data is stored.
- Data Lake uses the ELT (Extract Load Transform) process while the Data Warehouse uses ETL (Extract Transform Load) process.
- Data lake captures all kinds of data in their original form from various source systems whereas DW captured structured data and organizes them in schemas that are specifically defined for DW purposes.

- Data storing in big data technologies are relatively inexpensive then storing data in a data warehouse.

3.8 Conclusion

A Data Lake is a storage repository that can store great amount of structured, semi-structured, and unstructured data. The main objective of building a Data Lake is to offer a raw view of data. In chapter four, we stated all the important aspects of a DL, its principals and how to extracted value from a Lake. in addition, we mentioned the major components of a DL Architecture: Data Ingestion, Data storage, Data Governance, Data Security, Data Analysis, etc.

So, the design of Data Lake should be driven by what is available instead of what is required. Data Lake reduces long-term cost of ownership and allows economic storage of files. The major risk of data lakes is security and access control. Sometimes data can be placed into a lake without any oversight, as some of the data may have privacy and regulatory need.

In summary, getting the right platform, loading it with the right data, and organizing and setting it up for self-service with skills and needs-appropriate interface are the keys to creating a successful Data Lake.

Chapter 4

Data Ingestion

We began chapter four with an introduction to dive in to the fine details of data ingestion. Here we have presented a detailed definition of data ingestion, the challenges that come with it, its architecture and its parameters. Since in our thesis project we will be creating and manipulating data pipelines, we have presented all the required information concerning data pipelines. Immediately, we have a structured description of the CDC procedure. Finally, we have listed all the essential and highly used tools to ingest data into a DL. Finally, we closed this chapter with a conclusion.

4.1 Introduction

It is no doubt that a Data Lake is built up in various components and that's what make it an enormous and complex ecosystem. For our thesis project, we've selected a precise segment from this vast ecosystem and have decided to break it down and examine it thoroughly. The fragment we've decided to break down is Data Ingestion. The framework is about moving data and especially the unstructured data. Data ingestion is also about collecting information from multiple sources and putting it somewhere it can be accessed. This process is the beginning of the Data Pipeline.[19]

Data ingestion problem handles numerous volumes, speeds and formats of data. Like mentioned in the previous chapters, this data can flow in batches or streams. Traditionally the data ingestion processes were frequently discussed through ETL tools via created pipelines, but this process is slow and not time-sensitive.

The quality of your ingestion process certainly determines the quality of your data. In simpler words, ingest your data incorrectly you'll end up with valueless data, misleading analysis results thus ending in jeopardizing the value of your data all together. On the other hand, if you ingest your data correctly and take in full consideration the data preparation stage automatically your data arrives in the Data Lake on time, with the correct fidelity and ready for data wrangling and analytic use.

The ingestion framework consists of two components, data collector and integrator. Data collector is responsible for collecting or pulling the data from a data source and transporting it to the data pipeline. It is a layer when components are broken so that analytical capabilities may begin. Data Integrator is accountable for ingesting the data into the Data Lake. The data collector and integrator components can be flexible as per the big data technology stack when it comes to the implementation of the two. As mentioned in the previous chapter, data can be ingested either in real-time or batch mode. Real-time mode means that the usage of the "Change Data Capture" (CDC) framework is applied. The transaction logs that are being replicated in the data lake are being read by the CDC framework.

4.2 Data Ingestion Challenges

There are a few challenges that can impact the data ingestion layer's pipeline[12]:

- Guaranteeing the legitimacy of the data so that it follows the accurate format is necessary. The difficult relationship between data quality and business needs exists. When data becomes tremendously vast, the task becomes costly and this is where mistakes happen.
- The processes of data ingestion can be fragmented and this can lead to duplicates. This will surely lead to results which overlap and data drifting due to different departments dealing with problems in their own personal manner and devices. If the source data is poorly managed and documented it may result in difficulty trying to bend data managed by third parties to your personal benefits.
- If the future of the ingestion pipeline is not considered, including the validation of data, which is often a neglected it can result having issues connecting with external systems. This can cause delays, increase costs and frustrate end users.

4.3 Architecture of Big Data for Ingesting data

The Architecture of Big Data is consisted of 6 unique layers, which secures the flow of the data and also helps design the data pipeline with the various requirements of either the batch or stream processing system. Below, Figure 4.1 represents a precisely structured image displaying how the architecture of big data sits when ingesting data. The arrows in that image describe how data is flowing in and out from different layers. In addition, we've briefly explained what each layer is responsible for and their main tasks.[8]

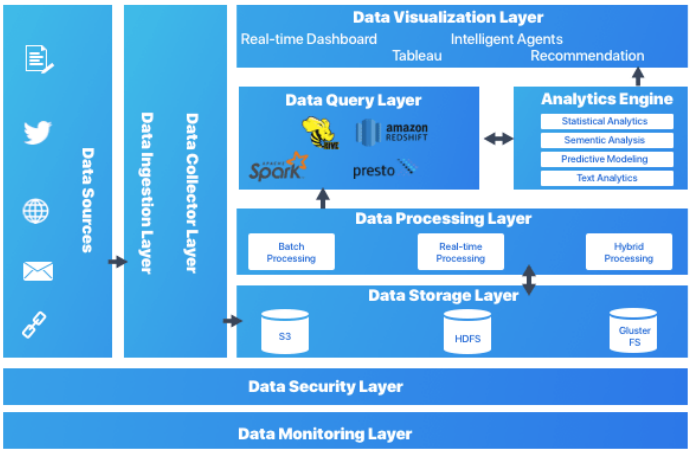

Figure 4.1: Architecture of Big Data Ingestion [25]

4.3.1 Data Ingestion Layer

This is the layer where all the data is arriving from numerous sources to begin its transformation. In other words, data is placed into bulks and ordered in a way to make data flow effortlessly into the other layers.

- **Batch ingestion**
  Since data sets are massive, its most likely that data must be processed in batches or chunks to be able to filter, aggregate and prepare data for the next step, analysis. The main steps of these jobs involve reading data from source files, processing them and writing the output to new files. Options include running custom MapReduce jobs in Hadoop cluster, or using Scala, Python and Java programs in Spark cluster, Hive etc.

- **Real-time ingestion**
  If data is flowing continuously, it means the solution includes real-time sources. Automatically the architecture must contain a way to capture and store real-time messages for stream processing. So, incoming messages are dropped directly into a folder ready to be processed in a simple data store. Options include Azure Event Hubs, Azure IoT Hubs, and Kafka. For real-time ingestion mode, a change data capture (CDC) system can suffice

the ingestion requirements. The change data capture framework reads the changes from transaction logs and are replicated in the Data Lake.

4.3.2 Data Collector Layer

In this Layer, the data components are broken down so that the analysis can be applied. This layer is also focused on how to transport the data from the ingestion layer to the rest of the data pipeline and further into the layers.

4.3.3 Data Processing Layer

The main focus in this layer is to specialize in the data pipeline process. After receiving all the data from the previous layers, it is now time to process the new established data. In this layer, the destination of data is clarified, the data flow is classified and the first stages of analysis may happen.

4.3.4 Data Storage Layer

One of the main challenges of storage is when the size of the data you are dealing with becomes extremely large. An obvious solution to this problem would be data ingestion patterns and there are plenty more solution. The main question asked in this layer “where to store such large data efficiently?”.

4.3.5 Data Query Layer

Active analytic processing takes place in this particular layer. The aim of this layer is to collect the data value and transform it in a manner to make it useful for the upcoming layer.

4.3.6 Data Visualization Layer

The presentation layer is perhaps the most significant and important tiers. The data pipeline users may feel the value of data. The foremost purpose is to grab the full attention of people and present the results in an understandable, comprehensible and readable manner.

4.4 Data Ingestion Parameters

Data ingestion has three essential parameters.

4.4.1 Data Lineage

It determines the life cycle of data – its purpose to display the full data flow, from the beginning to end. Data lineage is a procedure that consists of understanding, tracking and visualizing data as it comes in from data sources all the way to the consumption layer. All the modifications and transformations that data underwent is included. It also includes how, what and why the data was transformed. Data lineage permits businesses to:

- Keep tab on the mistakes in data processes.
- Uses process modifications with minor risk.
- Completes system migrations with assurance.
- Combines data discovery with a complete understanding of metadata, to create a data mapping framework.
- Data lineage assures users that their data is arriving from a secure source, transforms it in a suitable manner and loads it into the specified location. Since strategic decisions rely on precise information then it's no doubt that data lineage is a vital role for them.
- Data becomes practically unbearable or very expensive and time consuming to verify if the data processes isn't being tracked accurately.

4.4.2 Data Velocity

It takes in consideration the speed at which data flows from numerous sources such as web clicks, machines, networks, media sites, social media. This measure can either be enormous or constant.

4.4.3 Data Frequency

It defines how regularly we ingest a certain data-set or the rate in which data is being created. Data can be processed in real-team which is continuous ingestion or batch which indicates ingestion in specific time intervals. In the real-time, data is transformed directly. While, in batch processing, first data is deposited in batches and then moved.

- **Data Size:** It indicates the volume of data which is created from several sources. Whether it's the normal or maximum size of block in a one ingestion procedure.
- **Data Format:** Data can come in countless formats. So, incoming data must be determined whether the data flow's format is either structured, semi-structured or unstructured.

4.5 Data Pipeline

A source, processing steps and a destination are the three elements that make up a data pipeline. In various data pipelines, the end point or destination may be called a sink. Using data pipeline has allowed the flow of data from applications to data warehouses, from Data Lakes to analytics database etc. A simple example, data pipeline on many occasions may share the same source and destination since data pipelines are mainly concerned about modifying the data set. In simple words, between point A and point B, any data that is processed between those two points, rest assured that there is an existing pipeline. A data pipeline is a system that helps filter data and formats it to more efficiently helpful insights without any extra irrelevant data points. Data ingestion, storage or analysis all can take place in a data pipeline. The main aim throughout the usage of data pipeline is to offer precise data, to make it easier to report, analyze and use. Since pipelines offer reduced data noise, tailored, organized data and provide only information required to accomplish goals, these advantages clear out blurry visions to business intelligence and enterprises. Data pipeline is consisted of multiple steps which aid in moving raw data from the source to the destinations. A source might be for example a data base whereas the destination can be typically a Lake or Warehouse.

4.6 Data Pipeline Elements

According to Figure 4.2, these are the main components of a typical data pipeline and how large data sets are prepared by them ready for analysis.

Source

Data pipelines extract data from these places. These sources can be anything from RDBMS, social media and even IoT device sensors.

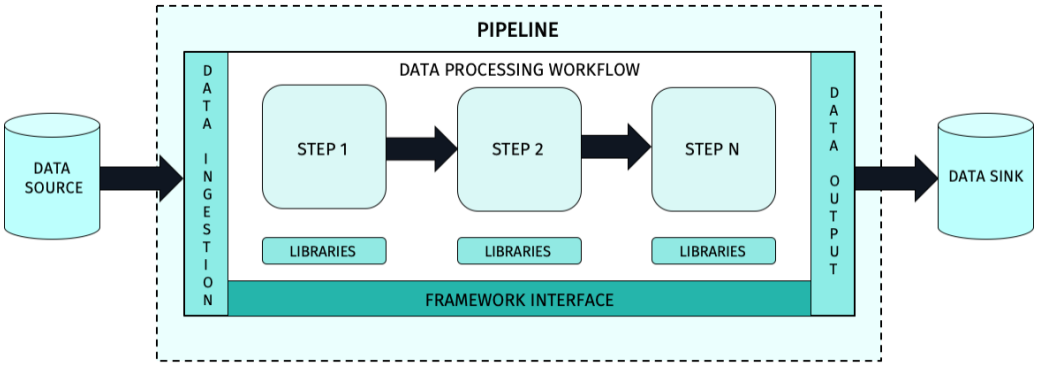

Figure 4.2: Architecture of Data Pipeline [31]

Destination

This is the final stop for data pipelines, this is where all the data is dropped after is has

been extracted. The destination can be either a Data Lake or a Data Warehouse. Data is stored in these places waiting for analysis.

Dataflow
While data is traveling from the source to the destination, data undergoes many transformations. This movement of data is called data flow. ETL is the one of the most common data flow approaches.

Processing
Processing includes extracting data from the source and transporting it to the destination. How data flow is implemented is decided by the processing component of a data. For example, choosing between batch or stream processing when extracting data.

Workflow
Workflow handles the sequencing and the order of jobs in a data pipeline and their reliance on each other. Reliance and sequencing both decide when a data pipeline runs.

Monitoring
Anything that needs to be perfected needs constant monitoring. In the data pipeline the monotorization is done to assure the accuracy of data and monitor any data loss. If not being well monitored we can lead to lose of data and even worse scenario concluding incorrect results. A pipeline is in fact monitored for speed, efficiency and mostly when the data is growing

4.7 Data Ingestion Tools

4.7.1 Apache Sqoop

Sqoop is a tool that aims to transfer data between HDFS (Hadoop storage) and relational database servers.
Sqoop uses MapReduce jobs to import and export the data. Data import and export provides parallel operation as well as fault tolerance.

Sqoop Architecture

SQL has designed all the existing Database Management Systems by its standard. Yet, every DBMS is different with respect to dialect to some degree. Due to this variance, many issues have been imposed when it comes to transferring data across the systems. This is where Sqoop connectors save the day and overcome these challenges. Sqoop connectors have facilitated the data transfer between Sqoop, Hadoop and external storages. Sqoop has connectors for collaborating with MySQL, PostgreSQL, Oracle, SQL Server, and DB2. All these connectors have the ability to interact with its connected DBMS[4].

Figure 4.3 is a simple illustration to display how Sqoop connectors create pipelines between the data source and target destination.

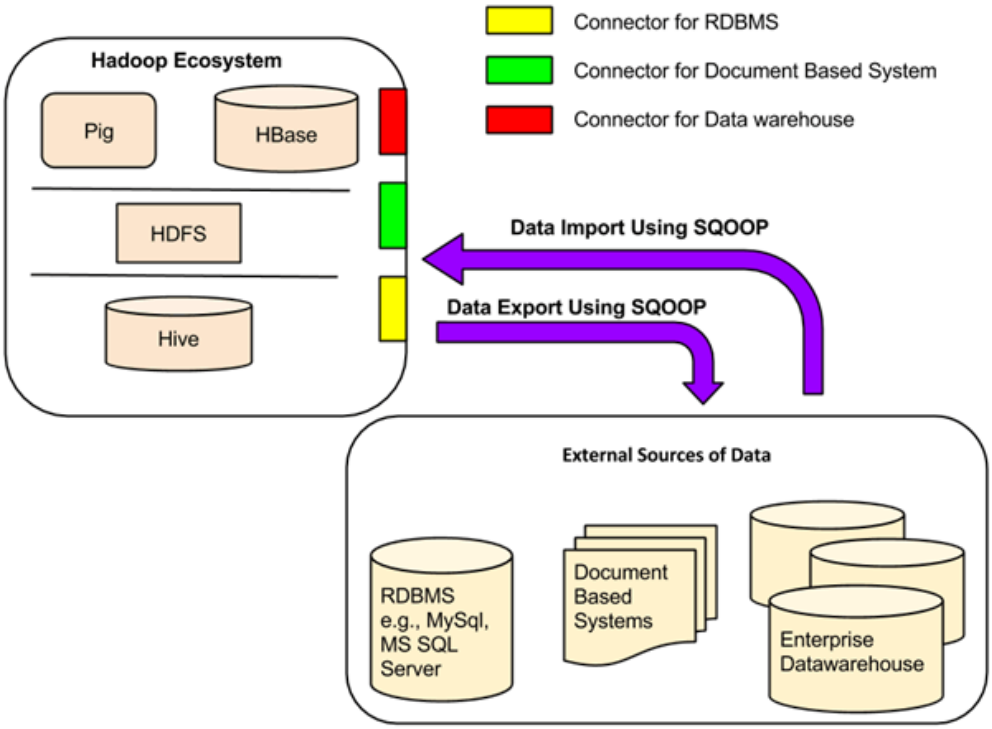

Figure 4.3: Architecture of SQOOP [6]

1. **Sqoop Import tool**

   To simplify the procedure of the import tool, it's done in two steps as shown in Figure 4.4.

   a) Sqoop self-reflects the database to collect the essential metadata for the data being imported

   b) Sqoop submits the mappers to the Hadoop cluster. This is the specific procedure that ensures the data transfer using the metadata taken in the previous step.

   HDFS directory carries all the data imported from RDBMS. There alternative HDFS directory such as Hive, HBase etc. Originally, all imported files are separated by commas and new line indicate record separator.

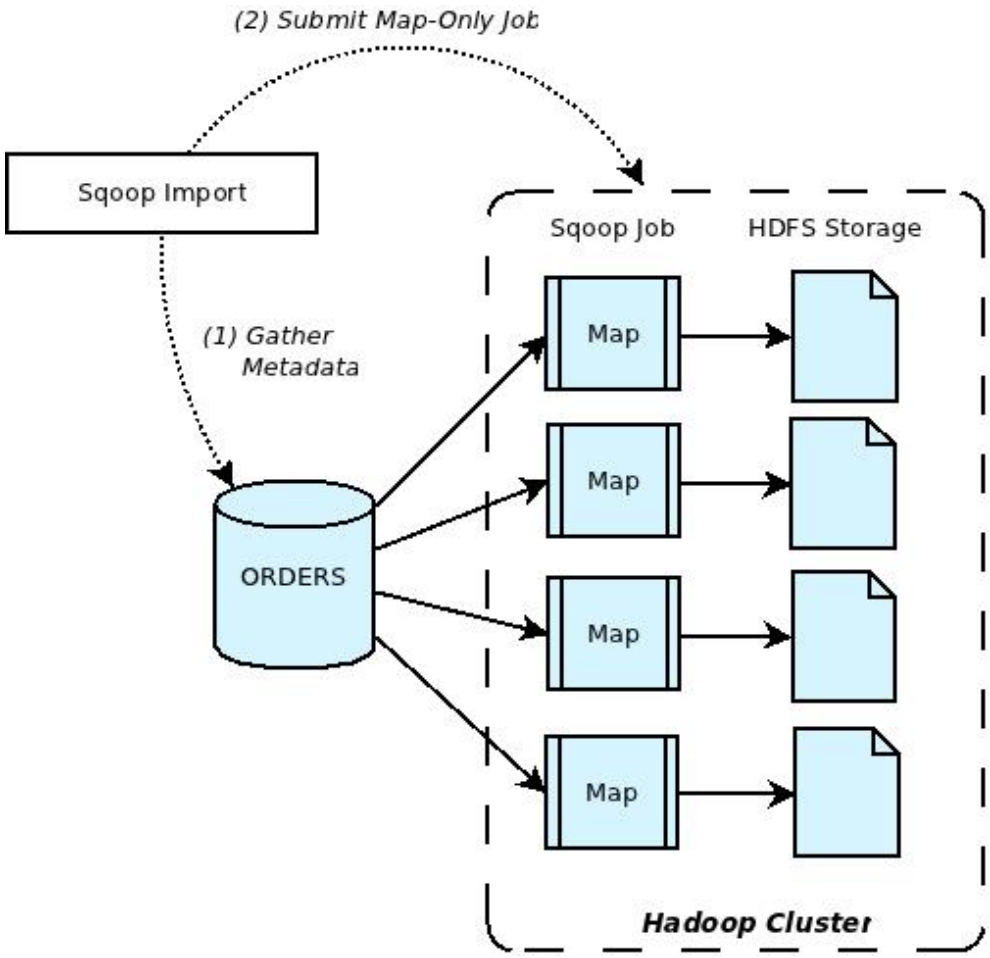

Figure 4.4: Sqoop import tool [3]

2. **Sqoop Export tool**

   The Sqoop export works in the same manner as the import tool. The aim of this tool is to export the data set from HDFS back in to the relational databases.
   The Sqoop Export tool exports the set of files from the Hadoop Distributed File System back to the Relational Database.
   Export is completed in two steps as displayed in Figure 4.5.

   a) The first step is to retrieve the metadata information from the database, in this case HDFS.

   b) Second step to transporting data.

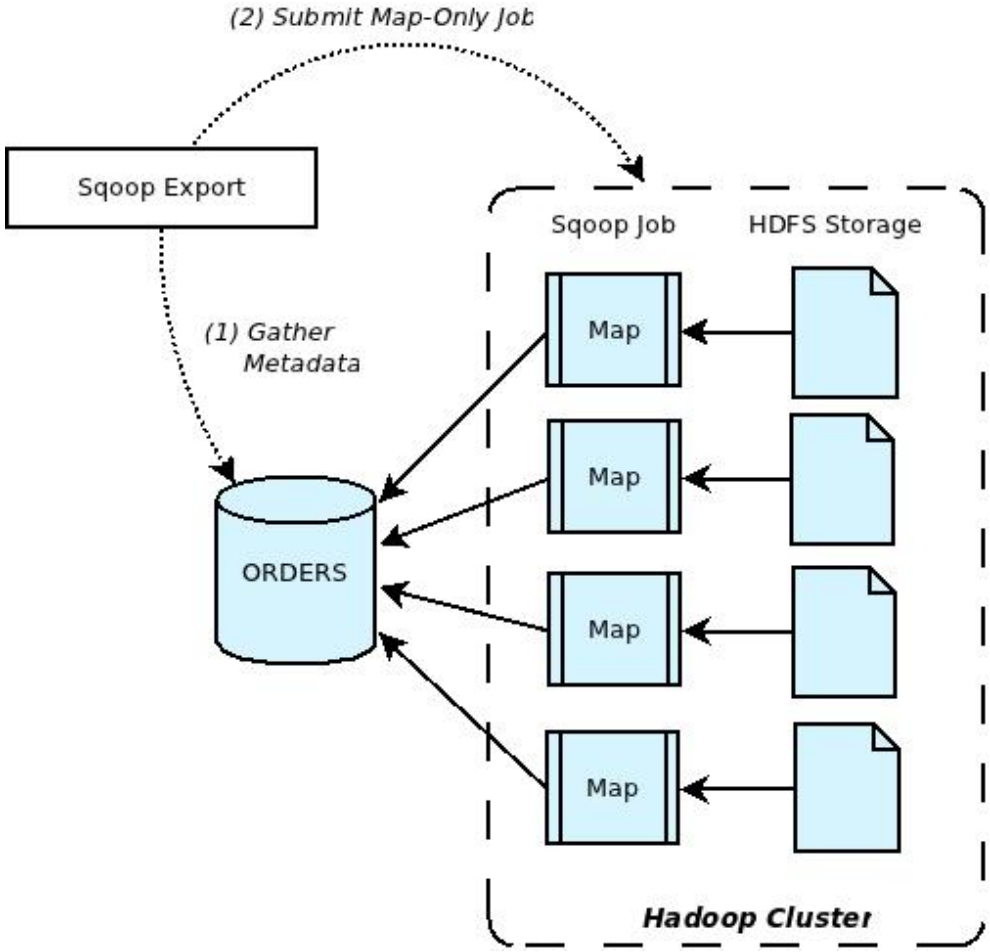

Figure 4.5: Sqoop export tool[1]

Connectors

As mentioned before, Sqoop uses connectors to facilitate the connectivity with external systems and the utilization of import and export tools. After a connection has been installed, Sqoop is able to use these plugins to effectively transfer data between Hadoop and the external system. Other than that, Sqoop has given the chance to other companies to develop their own connectors that can be plugged into Sqoop for their personal benefits.

4.7.2 Apache Kafka

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications. It uses a "distributed, Partitioned and Replicated Commit Log Service ". The data streams are partitioned and distributed over a cluster of machines, permitting data streams larger than the capacity of a single machine to be stored without any disruption.[4]

Kafka Architecture
Kafka can be used as a cluster on multiple servers, where the cluster store record streams under different categories known as topics. Each record has a key value and is used for two types of application:

1. To build data streaming pipelines that move data between systems and applications.
2. To build application that react and transform the incoming data streams.

Kafka provides five APIs (Application Programming Interface) which are essential for its design:

- Producer API: Allows applications to publish streams of records to one or more topics.
- Consumer API: Permits applications to contribute to one or more topics and process. records to them.
- Streams API: Allows applications to act as stream processors, by absorbing input streams from multiple topics and releasing altered data to multiple topics.
- Connector API: For building and running of reusable producers or consumers so that Kafka topics can be connected to existing data applications/systems.
- Admin API: For managing and reviewing the topics, brokers and other Kafka objects.

Kafka topics
All Kafka records are prearranged into topics. Producer applications are responsible to write data to topics whereas consumer applications are responsible to read from topics.

Partitioning
In a cluster, topics are separated into partitions, and the partitions are replicated across brokers. All messages that hold similar key will be sent to the same partition. Apache Kafka has no limit on the number of partitions that can be created.

Kafka brokers

Since Kafka is a distributed data infrastructure, a broker is a node that can be replicated on a network so the collection of all these nodes works together as a single Kafka cluster. Kafka brokers are deliberately kept very simple. They are accountable for writing new events to partitions, executing reads on existing partitions, and duplicating partitions between them.

Replication

In Apache Kafka, in the partition layer is where replication is implemented. Each topic owns a configurable replication factor that determines how many of these copies will exist in the cluster in total.

Kafka Consumers

The consumer is an external application that reads messages from Kafka topics and are allowed to read from any offset point they choose and does some work with them like filtering, aggregating, or enriching them with other information sources.

4.7.3 Fluentd Data Collector

For a unified logging layer exists and open-source data collector known as Fluentd. It permits the action to unify data collection and consumer for improved usage and comprehension of data. Many organizations use Fluentd and Fluent Bit to collect, process, and transport their data from cloud infrastructure, network devices, Kubernetes, and plenty other sources.

Fluentd functionality

Fluentd scraps logs from a specified set of sources, processes them by changing into a structured data format and then sends them to other services like Elasticsearch, object storage etc. The following steps explain how Fluentd work:

1. Fluentd extracts data from many data sources.
2. Transforms data format in to structured data and tags it.
3. Based on matching tag, it forwards the data to numerous destinations.

Fluentd Architecture

The most popular architectures: Forwarders and Aggregators, Side car / Agent deployment and Network device aggregator.

1. **Forwarder and Aggregator:** Deploying in forwarder/aggregator pattern is one of the most popular patterns for Fluentd/Fluent Bit. A lightweight instance deployed on edge, generally where data is created, such as Kubernetes nodes or virtual machines. Slight

processing is done by these forwarders and then use the forward protocol to send data to a much heavier instance of Fluentd or Fluent Bit. The filtering and processing are done by the aggregator which is the heavier instance before routing to the appropriate backend(s).

The Table 4.1, below represents the pros and cons of using a Forwarder and Aggregator in Fluentd architecture.

Advantages	**Disadvantages**
• Minimal usage of resource on the edge devices • Aggregator is scalable • Easy to add more backends	• Required stable and reliable resources for an aggregation instance

Table 4.1: Aggregator Advantages and Disadvantages

2. **Sidecar / Agent deployment:** This model uses deploying Fluentd and Fluent Bit on edge which is similar to the forwarder deployment. The only difference is that data isn't sent to an aggregator, instead it's sent directly to the back end. This method is efficient if there is one single back end needed to send data to and is used by cloud giants such as Microsoft, Google, and Amazon.

   Table 4.2, below represents the benefits and inconveniences of using a Sidecar and Agent deployment in Fluentd architecture.

Advantages	**Disadvantages**
• No aggregator is required.	• Difficult to modify configuration across a fleet of agents (E.g., adding another backendor processing) • Difficult to increase end destinations if required

Table 4.2: Sidecar Advantages and Disadvantages

3. **Network Device / Syslog aggregator:** Both Fluentd and Fluent Bit are Cloud Native Computing Foundation (CNCF) projects. Syslog is one of the most commonly used input that is included in Fluentd and Fluent Bit. Users that intent to capture all the logs and route to security-focused back ends, deploy pure aggregators to do it. These aggregators can also contain logic to redact specific messages or process messages in a more practical way for security applications in end destinations.

   Table 4.3, below represents the advantages and disadvantages of using Network Sevice/ Syslg aggregator in Fluentd architecture.

Advantages	Disadvantages
• No agents required; Primarily read from Syslog. • Add processing after data is sent, such as IP redaction, and scale independently.	• More processing might be required depending on the input. • Troubleshooting might be more involved with black-box network devices

Table 4.3: Network Device Advantages and Disadvantages

4.7.4 Data Integrator

4.7.4.1 Change Data Capture (CDC)

Change data capture keeps tabs on modifications that occur in data sources and automatically transfers those changes to a target dataset.[5]

- CDC is often used to replicate data between databases in real-time.
- CDC instantly and automatically syncs databases as soon as the data source changes and constantly tracks changes in a source database.
- Uses stream processing to ensure instant changes.

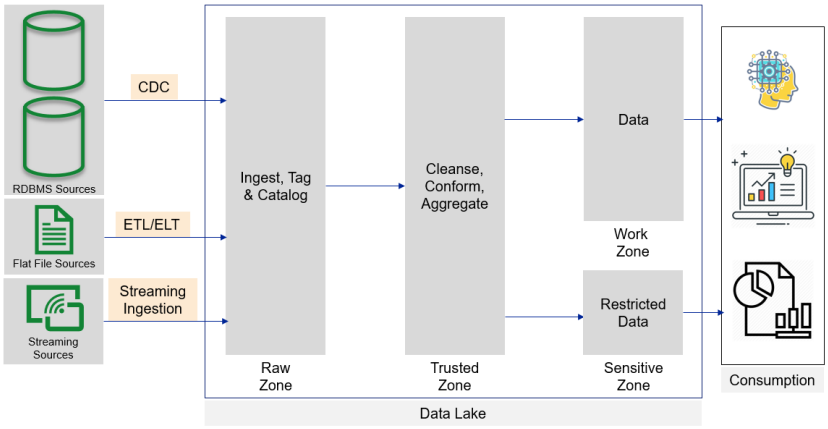

Figure 4.6: Architecture of CDC approach [32]

CDC is an approach to data integration that is based on the examination, capturing and delivery of the change to data source. Rendering to Figure 4.6 that represents the architecture of CDC, it works by capturing alterations that have been created in a database level of the rows and proceeds on by replicating them to another place, database or data store. CDC pipelines can be useful for data replication, such as to a Data Warehouse or for ETL jobs..

In a modern data architecture, we can endlessly ingest CDC data into Data Lake in the aid of data pipeline. CDC can support the analysis of streaming logged data. Indulging large amounts of data in to Data Lakes can sometimes causes congestion problems and the most common solution would be the help of CDC. CDC just updates the changes to the Data Lake. For the record, numerous establishments have used DL's over ETL platforms as the Data Lake environment is less expensive.

The most problematic part of the Data Lake is preserving it with current data. CDC can help to save computing and network costs, especially in the case of cloud targets. With support for technologies like Apache Spark for real-time processing, CDC is the underlying technology for driving advanced real-time analytics.

Steps to load data from source to destination

a) Database Dump
   Database dump is a simple solution that can be used when we export database with petite sizes and later on import them to new data marts, lakes or warehouses.

b) Change Data Capture (CDC)
   CDC comes in when we reach a certain size where SQL dump is no longer the suitable solution in meeting our data needs . CDC will only capture the change in the data. It is a highly efficient technology for reading the changes made to a data source and applying those to the destination. CDC records, writes, deletes and updates events. It copies numerous tables in their complete format from a source database into the ultimate database [5].

Types of Change Data Capture techniques

CDC can be placed into two categories[17]:

a) Query Based
   At the source, the execution of SQL statements is obligatory. The implementation of CDC, involves an impact on the performance of the source from which the data is extracted, this requires performing an I/O operation to the database by traversing an entire table.

b) Log Based
   The CDC process approach involves reading log files of the source database to identify the data that is being created, modified, or deleted from the source into the

target Data Warehouse. There are many techniques to manage change data capture processes. The top techniques are: timestamp, triggers, snapshot and log-based technique.

4.7.4.2 SAS Data Integrator Server

SAS Data Management

SAS Data Integration Server is a configurable and comprehensive solution that can complete a extensive variety of data integration necessities. It can:

- Access all data sources.
- Extract, cleanse, transform, aggregate, load and manage data.
- Import and export metadata functions, and build/execute ETL and ELT process flows.
- Support data warehousing, migration, synchronization, federation and provisioning initiatives.
- Support both batch-oriented and real-time master data management solutions.
- Create reusable data integration services in support of service-oriented architectures and data governance.

The transformation of Big Data into large opportunity with data integration, data governance, event stream processing and data quality technologies is all made possible with data management technology from SAS.

How SAS data management works

If data is collected automatically, it must be managed. As the volumes, formats and sources of data grow, the importance of real-time processing grows too, and the top priority should always be good data managing data. Below is a list the fundamental technologies of data management.

- Data access: It's the capacity to access information from any source. Important data exist in several places such as text files, databases, emails, data lakes, web pages and social media feeds. Excellent access technology allows you to extract useful data from any type of data storage mechanism or format that's available.
- Data integration: It's a process that combines different types of data to display integrated results. Through data integration tools, we can generate and automate steps to do those ETL and ELT procedures. The results of this is to benefit us to reach better decisions.
- Data quality: Data quality is about guaranteeing that the data is precise and practical for the requirements for which it is envisioned for. Data that is unfinished or does not

match its purpose will not be dependable which leads to problems throughout the organization and expensive mistakes.

Superior connectivity and data access with SAS
SAS comes with many benefits when it comes to its vast connectivity system such as providing connectivity that is available for both batch and real time to more data sources on more platforms. Also, file reader and writer are accessible for HDFS and supports Hadoop's MapReduce. Lastly, a consistent data definition across all data sources are provided due to the complete and shared metadata environment.

4.8 Conclusion

In conclusion, although data ingestion is not a simple process in writing and can be costly in establishing its infrastructure and maintenance over time, a well-written data ingestion process can help a company make decisions and improve business processes. In addition, this process makes it easier to work with a large number of information sources and allows easy access for engineers and analysts alike.
In Chapter four, we have recognized the architecture to ingest massive data. Also, the parameters of data ingestion. Correspondingly, we discussed the elements that compose data pipelines and finished off with a broad explanation of CDC approach and all the popular tools to ingest data, altogether displayed in a rational manner.

PART 2

Related Work

Chapter 5

Data Ingestion Solutions

The purpose of the information presented in this chapter it solemnly aimed to present all the genuine data ingestion solutions that were well-deserved to be presented. We've mentioned solutions such as: Apache Spark, Gobblin, Marmaray, Lambda. We've talked about their different architectures and pipeline and how they ingest data in different manners. At the end of this chapter, we've compared the different solutions, listed the missing elements in their frameworks, selected the approach we chosen to adopt and locked this chapter with a conclusion.

5.1 Apache Spark

Apache Spark is an extremely fast cluster computing technology. It is based on Hadoop MapReduce. Spark is built to cover workloads such as batch applications, iterative algorithms, interactive queries and streaming.
Spark can be defined as a data processing framework that has the ability to rapidly perform tasks on massive sets of data. It can also process data tasks whether on a single or numerous computers. The key to the big data and machine learning world is based on these two qualities.

5.1.1 Features of Apache Spark

Apache Spark holds the following features.

- **Speed** SRun workloads 100x faster in memory, and 10 times faster when running on disk.
- **Supports multiple languages -**allows the developers to write applications in Java, Scala, R or Python.

- **Advanced Analytics** Spark not only supports 'Map' and 'Reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

5.1.2 Ingesting Data from Files with Apache Spark

Below, we will explain what happens behind the scenes while ingesting files through Apache Spark. The example we will be explaining is the ingestion of a CSV file[21].

a) Figure 5.1 describe the first step in every spark application: connect to a Spark master and get a Spark session. For every Spark application, the first operation is to connect to the Spark master and get a Spark session.

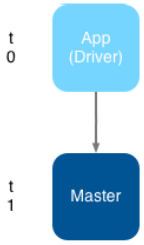

Figure 5.1: Connection between the Master and Session [22]

b) Figure 5.2 illustrate the next step: Ingestion. You need to ask Spark to load the data contained in CSV file. With all this said, masters recognize and rely on slaves or workers. In this illustration, you have three workers.

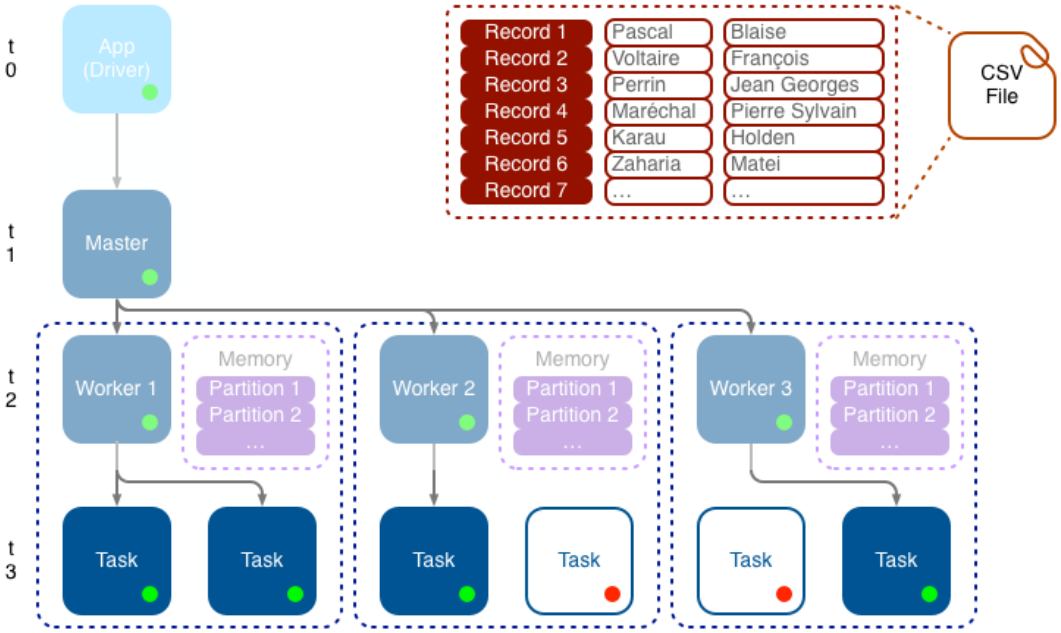

Figure 5.2: Workers [22]

Tasks are being generated based on the accessible resources. The slaves may create numerous tasks and allocate a memory partition to the task. The diagram above represents the running tasks that are with green dots, in contrast with tasks that are not running have a red dot.

Each task remains reading a chunk of the CSV file and the ingestion is done in rows and is stored in dedicated partition

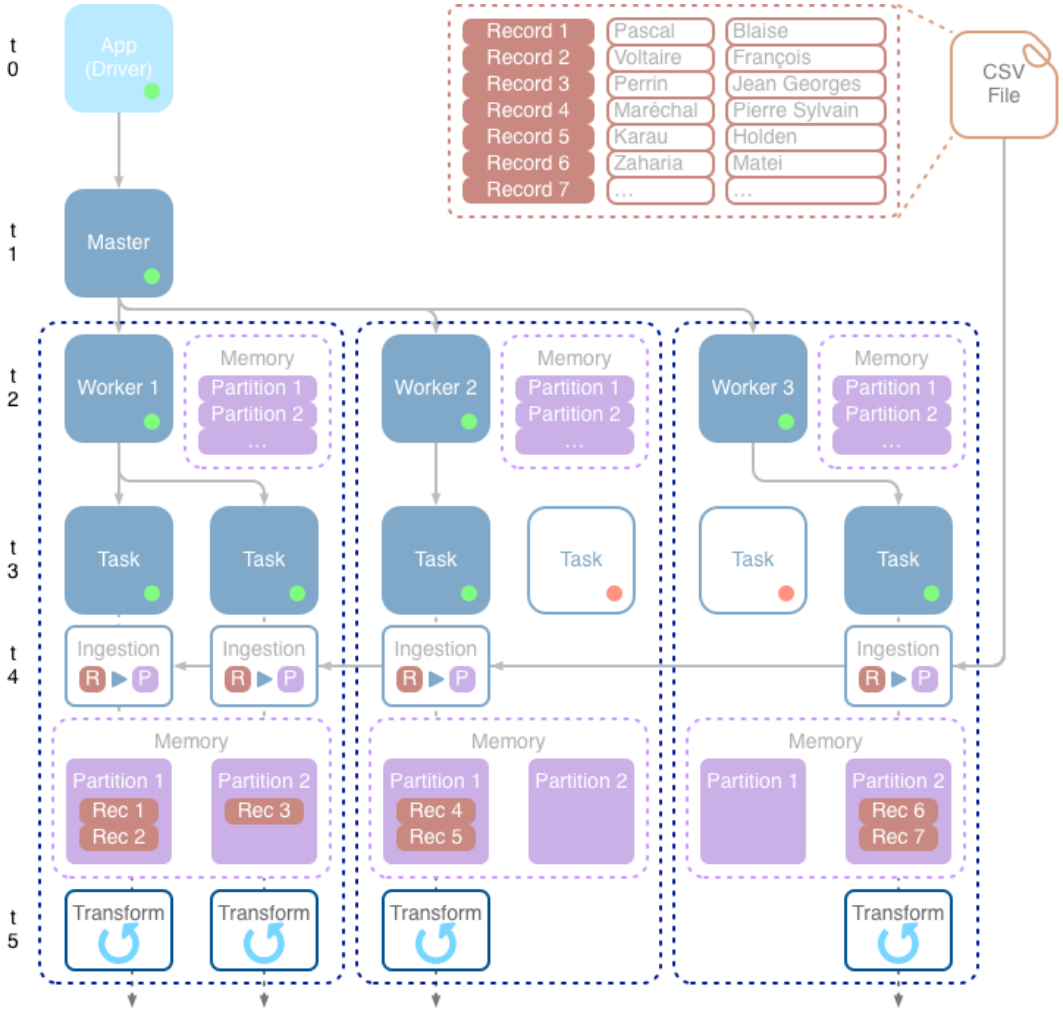

Figure 5.3: Ingestion procedure [22]

c) Lastly, Figure 5.3 above confirms that the ingestion is now taking place, each task is now loading some records into its own memory partition. You can also see the purple box which contains the records after the ingestion has been done. After the data has been loaded, the next step would be to process the records.

5.2 Apache Gobblin

Apache Gobblin is an integrated and generic data ingestion framework for Hadoop and it is one of the newest LinkedIn open-source product [23]. Gobblin's purpose is to unravel this problem by creating a centralized data ingestion framework that aims to facilitate and support the data ingestion from different data sources.

5.2.1 Challenges

Just with any approach Gobblin faces some challenges which are in the following list:

- The integration of differs source: Gobblin has managed to provide a large variety of adapters which are commonly used for data sources such as MySQL, Kafka, S3 and Google analytics, etc.
- Processing paradigm: The framework has managed to support standalone and scalable platforms, taking in consideration Hadoop and Yarn. The ability to run ingestion in batches or in streaming is all possible throughout the integration of Yarn.
- Extensibility: Since Gobblin has offered out-of-the-box adapters with the framework, it has become possible for data pipeline developers to integrate these adaptors and make it interpretable for other developers in the community.
- Self-service: standalone mode and flow deployment has permitted data pipeline developers to compose data ingestion and modifications in a self-serviced approach and test it locally.

5.2.2 Benefits

- Auto scalability
- Fault tolerance
- Data quality assurance
- Extensibility
- Handling data model evolution

5.2.3 Gobblin's Logical Data Pipeline

According to Figure 5.4 this is a graphical presentation of Gobblin's data pipeline to ingest data. Underneath are the definitions and roles of each component of this pipeline.

- **Source and Extractor**
  At the beginning of the job flow, a Gobblin job uses an adapter that connects to a

data source and Gobblin. A source is responsible for splitting the data ingestion work into chunks and also to create an extractor for each work unit. An extractor, connects to the data source and extracts data that is required.

- **Converter**
  A converter is the core construct for data transformation it is also responsible for converting both schema and data records when data is being pulled.
- **Quality checker**
  As the name suggests, is responsible for data quality checking. So, it determines whether the extracted data is suitable and can be published.
- **Data Writer**
  A writer is held responsible for writing data records to the sink it is connected to. The data can be published to numerous sinks such as HDFS, S3, Kafka etc.
- **Data Publisher**
  A data publisher is responsible for publishing the data extracted from a Gobblin job.

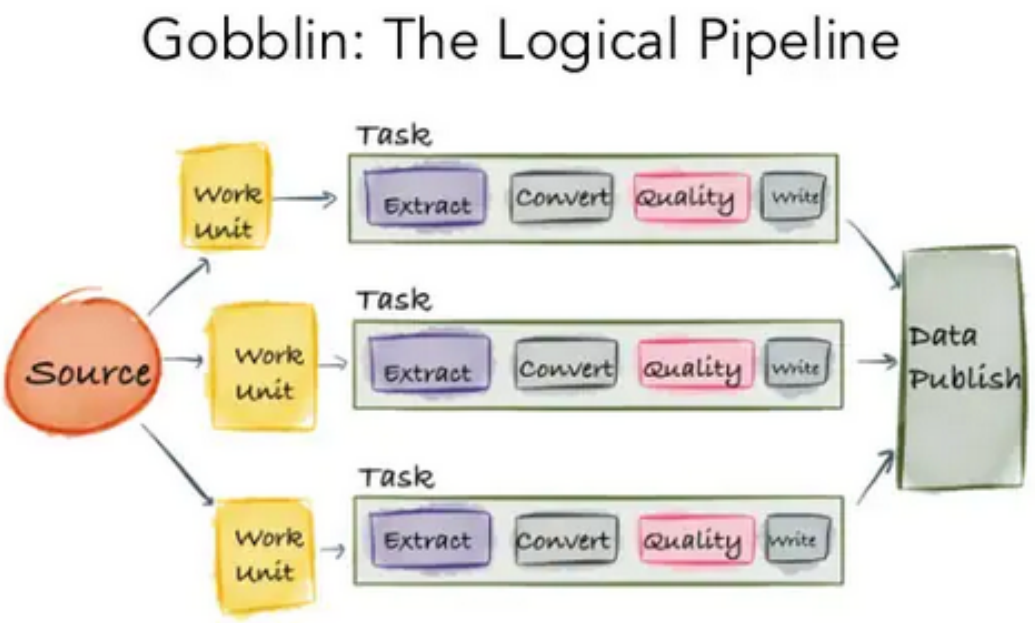

Figure 5.4: Gobblin's logical data pipelines [7]

5.3 Marmaray

Many of the fundamental building blocks and abstractions for Marmaray's design were inspired by Gobblin, a similar project developed at LinkedIn. Marmaray is a framework that supports the ingestion from any source and disperse to any sink. It is a plugin-based framework built on top of Hadoop ecosystem.

5.3.1 Benefits

- Produce quality schematized data through our schema management library and services.
- Ingest data from multiple data stores into our Hadoop data lake via Marmaray ingestion.
- Build pipelines using Uber's internal workflow orchestration service to crunch and process the ingested data as well as store and calculate business metrics based on this data in Hive.
- Serve the processed results from Hive to an online data store where internal customers can query the data and get near-instantaneous results via Marmaray dispersal.

5.3.2 High-Level Architecture

Figure 5.5 is the architecture diagram displays all the essential blocks that allow Marmaray the easy flow of jobs and add extensions to support new sources and sinks.

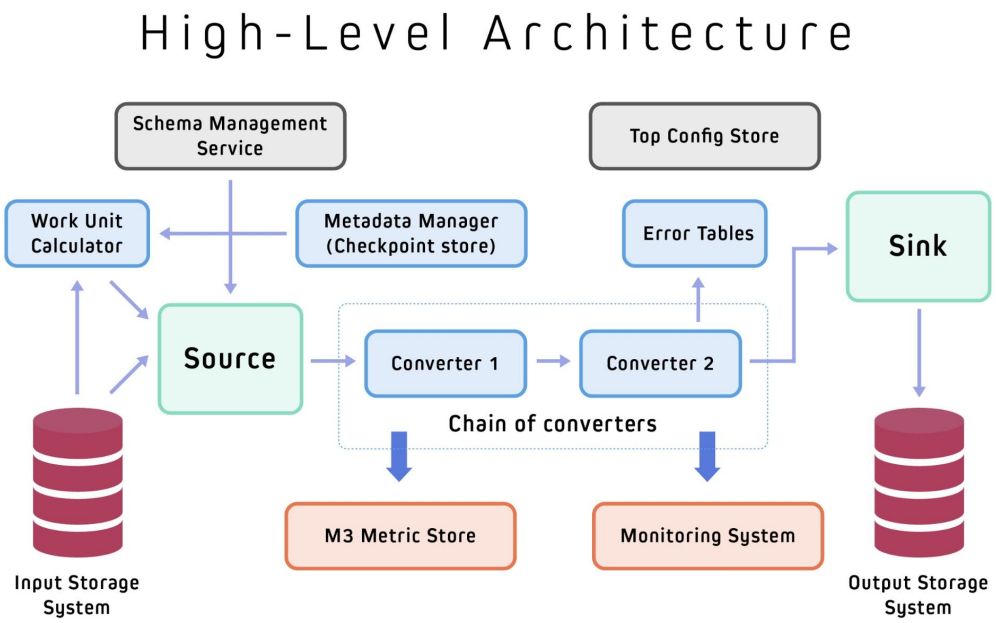

Figure 5.5: High-level Architecture of Marmaray [15]

Avro Payload

Avro Payload is the central component of Marmaray's architecture. Its where all the relevant metadata for data processing necessities is held.

Combining Avro data and running it on top of Spark's architecture offers us the ability to take advantage of Spark.

It's necessary that the ingestion sources have converters defines from their schema to Avro format in order to achieve "any-source to any-sink".

To help understand how Avro Payload functions, Figure 5.6 is an illustration that presents the process.

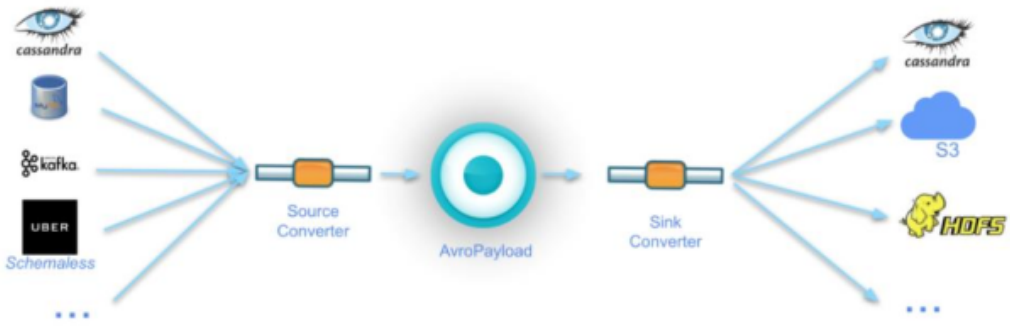

Figure 5.6: Avro Payload Process [15]

Data Converters

Earlier, data converters have been brought up. So, data converters aid data ingestion and dispersal jobs to achieve transformations from the data source to guarantee it is in the suitable format before writing the data to the targeted sink.

Error Tables

The intention behind error tables are to allow easy debugging of jobs and discard records that do not have suitable schema.

Work Unit Calculator

The Work Unit Calculator task is to calculate the total of data to process. At a very high level, the Calculator will inspect input source and the formerly used storage and then proceed to calculate the following work unit. For example, a Work Unit could be Offset Ranges for Kafka or a collection of HDFS files for Hive/HDFS source.

Metadata Manager

Metadata manager are responsible to ensure that Marmaray jobs have a persistent store to store job level metadata information. If recent execution of the job is successful, a job can effortlessly update its state during the execution and replace the old state. If that's not the case, no modifications to the state are accepted.

The illustration below presented in Figure 5.7 provides an image description of the Metadata Manager. Compromised in this figure, are essential elements of metadata manager: Storage, Manager and DAG components.

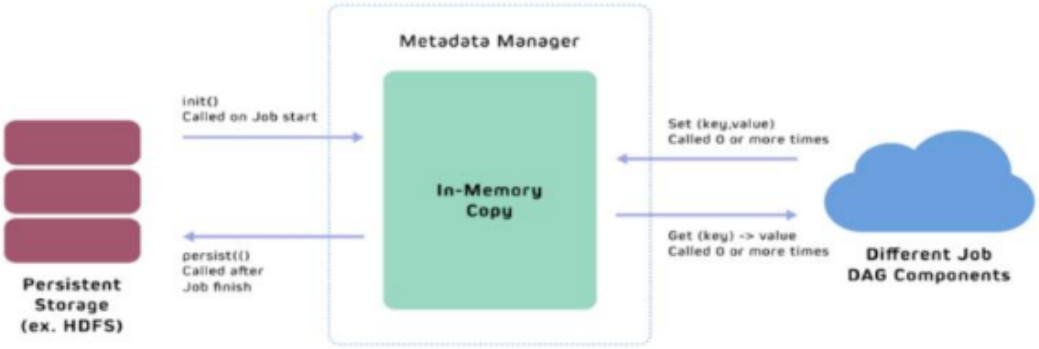

Figure 5.7: Metadata Manager [15]

Fork Operator

The aim of the Fork Operator is to split the input stream of records into multiple output streams.

ISource and ISink

The ISource holds all the required information from the source data for the suitable demanded work units. ISink holds all the essential information on the procedure to write to the sink.

5.4 Lambda

The Lambda Architecture is a deployment model for data processing designed for Big Data systems that involve data processing in near real-time. After ingesting data from thousands of IoT devices, it is then defined as a Big Data problem that needs to be solved. A technique is offered by the Lambda Architecture which is: building a single system that process near real- time streaming data and also provide the capacity to store and batch processed data using old data processing techniques[20].
The main components of Lambda Architecture are displayed in Figure 5.8

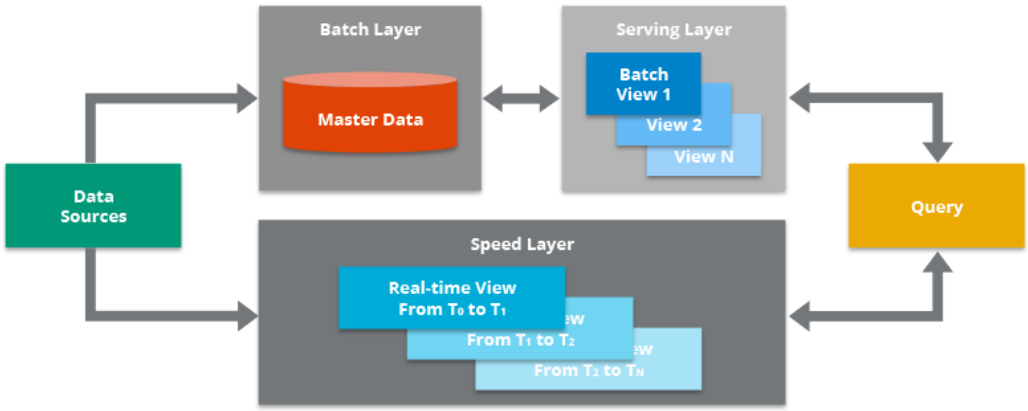

Figure 5.8: Lambda Architecture Components [33]

5.4.1 Lambda's Architecture

Data Sources

Data can be attained from numerous sources that can be compromised in this Architecture for data analysis. Apache Kafka works as an intermediator that can serve data in batch and speed layer of the Lambda Architecture.

Batch Layer

To prepare for indexing, all the data flowing into the system comes in batches and is saved in this specific component. Frequently, simple CSV format files are used. The HDFS is regularly used in order to ingest data as well as storing the data in a cost-effective way.

Serving Layer

In this component, the indexation of the newest batches occurs here to make queries more achievable by end users. Minimizing the time to index the data set is accomplished working in parallelized manner and this the most important obligation in this layer.

Speed Layer
This next layer balances the serving layer and that's by indexing the latest added data that are not completely indexed by the serving layer. This consists of both the data that is actually getting indexed by the serving layer and plus the new data flowing in after the actual indexing job started.

Query
The main responsibility of this component is to submit the end user queries to the serving and speed layer and joining the results. By accomplishing this task, it offers the end users a comprehensive query on all the data, compromising the newly added data and this offers a near real-time analytics system.

5.4.2 Principles of Lambda Architecture

- Fault-tolerant
  The main part of the Lambda pattern would be it is hardware, software and human fault tolerant. Lambda is designed to handle big data, therefore any of these errors can be an expensive issue to recover from. There is no place for data loss and corruption in this pattern due to the vastness of data. Another important component is human fault tolerance. Typical operational mistakes are the most common errors in everyday operations. The following mistake are the bugs that make their way into the systems over time. Therefore, the systems must be designed to deliver to and address these characteristics.

- Immutable Data
  The data should be stored in a raw format from the different source systems. The most crucial aspect is that data stored should be immutable. Data being immutable makes the system in general simpler and more manageable. This means that it shouldn't be converted or transformed indicating it is in its natural and raw format. By making it immutable, it handles errors created by human. To provide to important fast processing and performance, the data is often stored in a denormalized format.

- Re-computation
  Hence raw data is always accessible in the Data Lake, the possibility to cater to new necessities by running functions on the raw data is always available. Also, the data is stored in a schema-less structure due to relating data to a schema result in its own issue of re-computation, development and maintenance issues.

5.4.3 Advantages and Disadvantages of Lambda

Table 5.1 underneath displays the benefits and inconveniences of using a Lambda architecture.

Advantages	Disadvantages
• Batch layer of Lambda architecture manages historical data with the fault tolerant distributed storage which ensures low possibility of errors even if which is not beneficial in certain the system crashes. • It is a good balance of speed and reliability. • Fault tolerant and scalable architecture for data processing.	• It can result in coding overhead due to involvement of comprehensive processing. • Re-processes every batch cycle which is not beneficial in certain scenarios. • A data modeled with Lambda architecture is difficult to migrate or reorganize.

Table 5.1: Advantages and Disadvantages of Lmbda

5.5 Comparing different Ingestion Approaches

After examining the popular data ingestion solution, we created a table to present the various differences between the solutions. Take a look at Table 5.2.

Criteria	Apache Spark	Gobblin	Lambda	Marmaray
Latest stable release	3.0.0	0.15.0	2.0	Built and designed by Hadoop Platform team
Primary written in	Scala	Java	Java	Java
License	GPLv3(copyleft license)	Open source	Open source	Open source
Basic nature	Works well for ingesting the data from any source and load it to Hadoop, Spark mainly designs for big data, data science. It work with the streaming data, has various Machine learning library, can work on structured and unstructured data	Works well for ingesting the data from any source and load it to Hadoop also distribute the ingested data from Hadoop to various sinks	Works well for extracting data in a defined range from a data source and writing data to a sink such as HDFS	Designed to process huge amounts of data it uses batch processing to provide complete and accurate views of batch data simultaneously using real-time stream processing to provide online views of the data
Integrate data	Both(ETL/ELT)	ETL	ETL	ETL
Type of data	Both (Stream and Batch)	Both(Stream and Batch)	Both (Stream and Batch)	Both (Batch and Stream)
Streaming	YES	YES	YES	YES
Written data format	Parquet, JSON	JSON	JSON	JSON
Extent of ingestion	Local and Distance	Local	Distance	Local
Type of loading	Event driven	Event driven	Event driven	Event driven
Supported data format	Structured Semi-Structured Unstructured and Binary	Structured Semi-Structured Unstructured and Binary	Structured Semi-Structured Unstructured and Binary	Structured Semi-Structured Unstructured and Binary
Link to HDFS	Connected	Connected	Connected	Connected
Fault tolerance	Strong tolerance	Strong tolerance	Strong tolerance	medium tolerance
Notable users	Netflix, Yahoo, and eBay	Lorand Bendig	Yahoo, Netflix and Linked-IN	Uber and Michelangelo machine learning

Table 5.2: Comparison Table

After analyzing the different architectures, we managed to collect in a short period of time, we came to a decision that all previous solutions do in fact have the ability to ingest large and heterogeneous data in to a Data Lake. But after investigating their architectures intensively, component by component, we came to a realization that all these data ingestion frameworks were missing an essential component. What these frameworks do is, collect and extract data from the source. After that, in one data pipeline they begin to ingest data directly into Hadoop not once taking in consideration the organization or keeping everything in order. Right after that, they continue to load the data in to the chosen Lake and it slowly turning in to a chaotic mess of unorganized data.
These solutions haven't given any thought of the organization of the ingestion nor the data lake itself. By not precisely structuring the data pipelines before the ingestion occurs, they're just loading data in a disordered manner. By doing so, they are placing the Data Lake at an extremely elevated risk for it transforming into a swamp of data. A data swamp is essential data stored without organization and precise metadata to make retrieval easy. Unfortunately, a Data Lake can become a wasteland of data without clear organization. In many cases, copies of data or irrelevant data are gathered and dumped into storage.
Retrieving the data and then transforming data for analytics becomes a chore. This means if certain people are looking for specific data, they won't locate it. It happens quickly once companies convert to this storage method using ELT but do not clearly outline how to use it and what the outcome should be for it.

5.6 Conclusion

During our intensive research in search of a data ingestion framework to implement and append our features in order to enhance the overall procedure, we came a cross numerous solutions. The solutions that stood out for us were; Apache Spark, Gobblin, Marmaray and Lambda.
After comparing the solutions, taking in consideration our required needs to complete this thesis project, we came to a conclusion that we wanted to conquer Apache Spark as our foundation framework to ingest our numerous formatted data.
Apache Spark has realized fast evolution over the past years, becoming the most operative data processing and AI engine in enterprises today due to its speed, ease of use, and sophisticated analytics. Spark joins data and AI by simplifying data preparation at a massive scale across various sources.

PART 3

Contributions

Chapter 6

Suggested Data Ingestion Approach

In this chapter, we will present the roles within the framework of this thesis work to understand the process of big data ingestion in to a Data Lake environment. The information provided in this section: After introducing Apache Spark, we will explain why we have decided to implement and develop Spark as our data ingestion solution instead of the rest of the others that were available. In addition, a complete and thorough explanation about the added features that have been appended and how it will enhance the overall ingestion process.

6.1 Introduction

In the previous chapter, we explained in depth how Apache Spark ingests data into the Data Lake. A detailed explanation on how the masters and slaves work together in order to partition the data and load them into our sink (Hadoop).
Putting this complete architecture under the microscope, it is obvious that this setup can ingest structured, unstructured and semi-structured data, in batches or real-time streaming data. All of this can be stored in Hadoop's storage system as our Data Lake. Thus, resulting us in understanding that all types of data that are extracted from the source are directly sent in a single data pipeline and stored in the Data Lake. Even after it is stored in the Lake, there is no complete understanding of the data that is being ingested. Due to this reason we decided to add our own personalized touch to this architecture by implementing our own Data Classifier and Data Visualizer. After some intense research, we decided that these two elements were missing in this setup.
Our contribution to this architecture would be adding two new bricks to this setup. One before ingestion and that would be the Data Classifier and another after the ingestion is completed and that would be the Data Visualizer. The aim is to manage to classify the

flowing data before it is stored in our Data Lake and visualize it after to gain a better understanding of the data within.

6.2 Improvements to Apache Spark's Approach

We have presented our motives to why we chose Data Classifier and Visualizer as elements that will improve Apache Spark's ingestion framework throughout this simple example. In addition, a brief explanation on Sparks framework is also clarified.

The example that is displayed in Figure 6.1 below, is a graphical explanation of Apache Spark being incorporated in the ingestion of data. The first step during any ingestion framework, is to collect data from the source. In this case, we have all types of file formats that include: CSV, JSON, XML, etc. These files are then channeled into Spark and are broken up the data into chunks, called partitions. A partition is a collection of rows that sit on one physical machine in our cluster. A Data Frame's partitions represent how the data is physically distributed across your cluster of machines during execution.

After data is ingested by Apache Spark, then follows the next step which is the transformation of data. Any transformation that is applied on the data would be immutable meaning they cannot be changed once created. Since we are dealing with Data Lakes, we will not be applying any modifications to our data sets and this to guard the original raw format of the data. The alterations that will be applied are minimal, light and will not have any impact on the data itself. These modifications will only be applied when we need to manipulate the data in to the data pipelines, so they can facilitate the flow of data from the data source, through Spark and into Hadoop

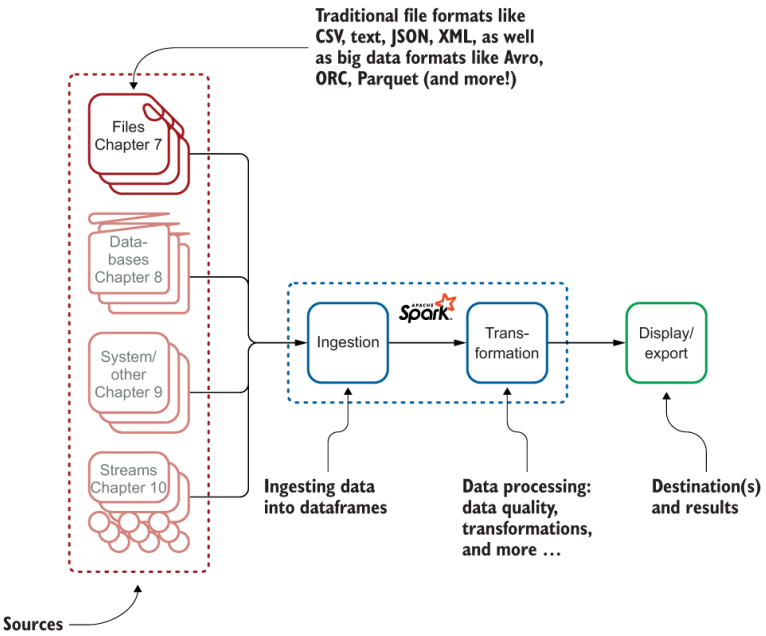

Figure 6.1: Apache Spark Ingestion Framework [22]

Next, the data is directly stored AS IS format into the Data Lake through one data pipeline. This means data will be loaded randomly into Hadoop without being placed into defined categories. The whole Lake will be undefined and be an ultimate mess and this risks the DL turning into a swamp. Due to this reason, we decided to implement a Data Classifier/Categorizer. We created separate data pipelines according to the formats of files. Basically, if we had five different formats that are ingested, we would create five separate pipelines in order to neatly ingest them and keep the Lake organized without touching the original format of files.
Finally, the data is stored in the Lake, Sparks framework is completed. This means anyone who wishes to understand what the DL is compromised wouldn't have a vivid image of the datasets due to the fact the human brain comprehends data that is neatly presented whether it is graphs, tables or pie charts. Spark doesn't implement Data Visualization to its framework and this is the reason why we decided to implement our second data brick that is Data Visualizer in to Apache Sparks Framework to enhance the clients experience when it comes to understanding the data within the Data Lake.

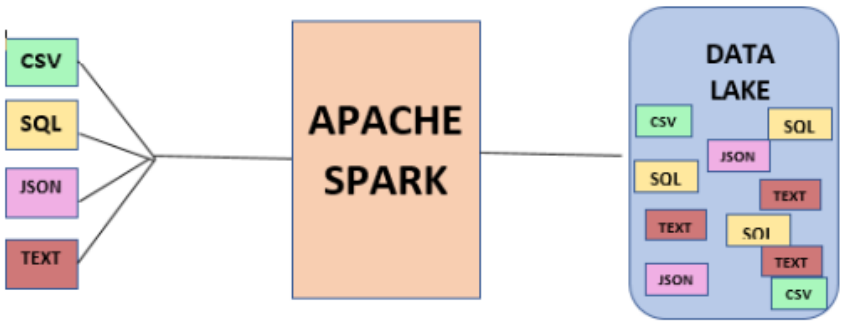

Figure 6.2: Apache Spark Work

Figure 6.2 above is a basic architecture on how Apache Spark ingests data. If looking closer at the architect, data is being extracted from the source and place in one pipeline, sent through Spark. After the data is partitioned into various clusters, it is then sent down one single data pipeline ready to be ingested. Lastly, it is loaded in the lake. Taking a closer look at the lake, you will notice that the ingestion has been completed in a random manner and the data is dispersed chaotically. If the first phase of the data ingestion is completed in a non-organized technique, this means the rest of the operations will follow this mess. Resulting in the lake turning into a total clutter, no control or order, making it difficult for the end user to locate data, traceability of data will become a challenge and major risk of the data lake turning into a swamp.

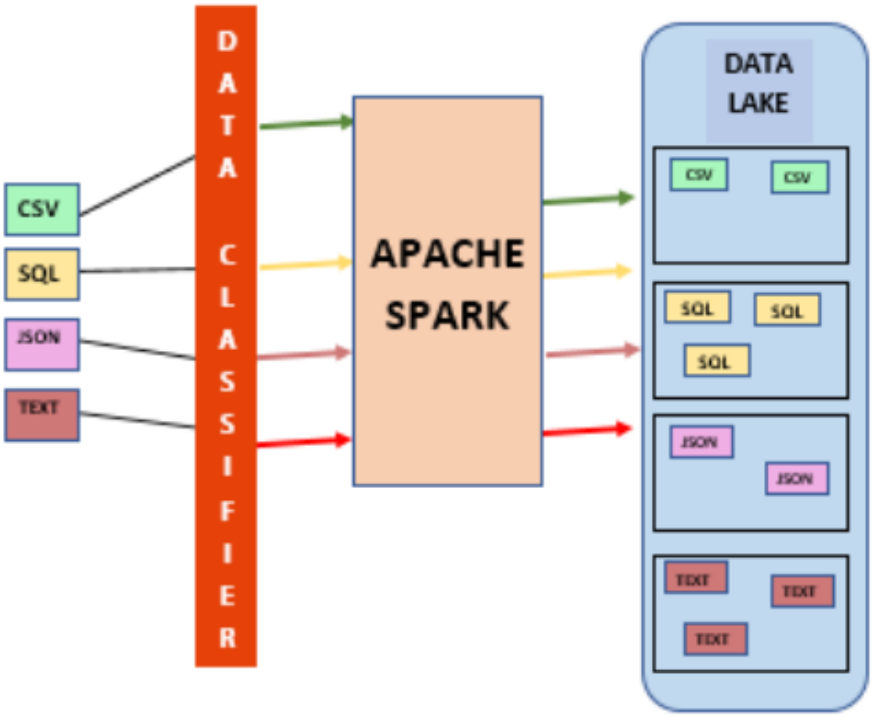

Figure 6.3: New Architecture adding the Classifier

Figure 6.3 above is the representation of our new architecture displaying our essential element. Our new added fragment to Apache Sparks framework is the data classifier. If you look at the architecture, and compare it with the old architecture and will see a huge difference when fixating on the overall organization of the lake.

The old approach extracted data from the source through one single pipeline, whereas our new approach, works by extracting from the source and passing the data through a classifier. The classifier then proceeds to place the data into distinct and specific categories ready to be parsed into Spark through designated data pipelines. In addition, the old approach ingests data through one pipeline and then are randomly appended to the lake in unordered method. This technique will place the lake in enormous risk of it slowly transforming into wasteland of data thus making it time consuming to locating data, traceability will be a massive challenge and just nightmare of an experience for the end-user. In contrast, our new approach ingests data in specified data, what that means that the data is placed in categories according to their format and they continue to be loaded in defined categories into the lake. Take a look at figure 6.3, you can see the lake, each category contains a specific format of files. So, not only are we organizing the data before the ingestion has begun, we are also organizing the data after it has been loaded into the lake. Resulting in a total order of the lake, less time consuming when searching for location of data and traceability has become more effective .

6.3 Data Classifier

A general definition of a data classifier would be that it is a process of organizing data into numerous categories that benefits the security and the overall usage of data. The foremost determination of a data classifier's process is to make your data effortlessly locatable and retrievable without needing to examine it again. As the previous definition suggests, the classifier aims to make it easier to find data throughout the usage of tagging which is found in metadata properties. To save storage costs and time to backup, this process has the ability to find and delete duplicated data. The process of classifying data isn't a task that can be completed overnight. It requires full attention to the smallest details and precise inspection. Data classification has numerous approaches and methods in the preparation and repository of data. Most of the time, these methods are applied on unstructured data that arrive in bulks from organizations and usually is an enormously tough task to manage. Figure 6.2 is a simplified illustration to facilitate understanding how a data classifier operates.

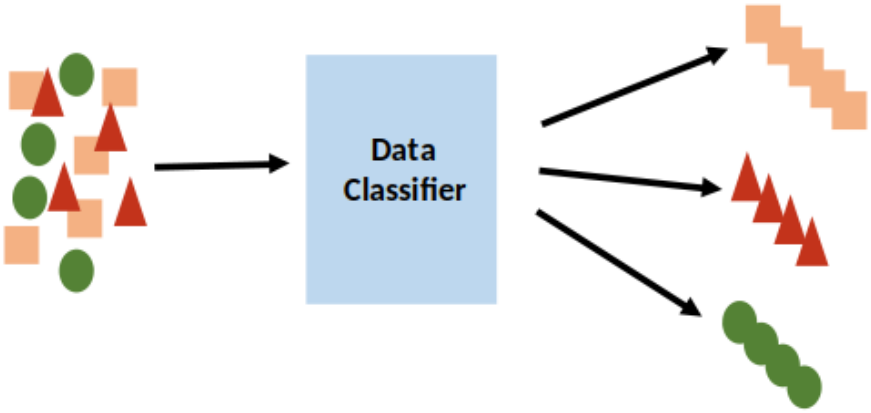

Figure 6.4: Data Classifier

6.3.1 Importance of Data Classification

Possessing a thorough understanding of the type of data we store and the location is the principal key into keeping your data's organization secure. The procedure to classify is to recognize and allocate pre-determined layers of sensitivity to different types of information. Classified Data is naturally categorized as either public or private. Public data can be read by anyone, anywhere and anytime and includes data such as birth certificates, marriage certificates, criminal records, etc. On the other hand, private data is data that can only be accessed through approval, and this includes Personally Identifiable Information (PII), Protected Health Information (PHI), etc.

6.3.2 Benefits of Data Classification

Most companies don't have knowledge to whereas the location of their highly sensitive data or how to accurately protect it. It has always been a tough problem to overcome since it's difficult to keep data secure, private and in compliance. By planning a data classification program, companies now have a huge advantage.

a) Support regulatory compliance: A data classifier aids in determining to find where data is located in the company, guarantees that suitable security controls are placed correctly. Data classification also ensures that the data is traceable and easy to find. The benefits that come with it are:

   - Guarantees that sensitive data is managed and used correctly for different regulations, such as medical, credit card, and personally information.
   - Helps in the capacity to uphold day-to-day compliance with all related rules, regulations, and privacy laws.
   - Ability to retrieve specific information with a precise timeframe, which helps meet newer compliance rules.
   - Expands the chance to pass compliance audits.

b) Increases business operation productivity and decreases business risks: Data classification has the capacity to ensure companies that they're data is successfully protected, stored and managed from the day it is created until its destroyed. These are the advantages that come with it:

   - Delivers better understanding and control over the data that establishments hold and share.
   - Access is allowed more affectively and data usage is protected.
   - Managing risks is now easier and that's by helping enterprises evaluate the value of their data and the effect of it being misplaced, stolen or altered.

6.4 Data Visualization

Data visualization is the graphical representation of data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets. Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from.
In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions. To get a better understanding of the different manners to visualize data, we've presented Figure 6.3 to reach the objective.[3]

Figure 6.5: Data visualisation[24]

6.4.1 Importance of Data Visualization

The practice of data visualization can benefit businesses recognize which aspects affect customer behavior; locate areas that require to be enhanced; make data more memorable for investors; comprehend when and where to place specific products; and predict sales volumes.[1]
Additional benefits of data visualization:

- the capability to absorb information rapidly, improve understandings and make quicker decisions.
- an amplified understanding of the following steps that must be taken to improve the company.

- an improved capacity to preserve the customers interest with data they can comprehend.
- an easy distribution of information that increases the opportunity to share insights with everyone involved.
- an increased ability to act on findings rapidly and, thus, accomplish success with greater rapidity and fewer mistakes.

6.4.2 Benefits of Data Visualization

Data visualization aids decision makers in numerous ways to improve data insights especially when it comes to considering business strategies and goals. The following are reasons why data visualization is extremely beneficial:

Quick action

The human brain comprehends visuals effortlessly compared to table reports. Data visualizations permit decision makers to be informed quickly of new data insights and take required actions for business growth.

Identifying patterns

Big volumes of data can offer many opportunities for insights when we envision them. Visualization allows business users to recognize relationships between the data, providing greater meaning to it.

Finding errors

identifying any errors in the data is simplified through visualization.

Understanding the story

The main purpose of the dashboard is storytelling. By designing your graphics in a meaningful way, you aid the audience understand the story in a single glance.

Grasping the latest trends

Discovering the latest trends in your business to provide quality products and identify problems before they arise is now possible when visualizing your data.

6.5 Adapting the Classifier and Visualizer in to Sparks Framework

As mentioned earlier, we've decided to implement the Data Classifier and Data Visualizer. The following steps describe the procedure of classifying the data.

a) **Extracting numerous formats of files**
We limited the number of formats to ingest: CSV, JSON, SQL and TEXT files. We began by selecting numerous of files that we were required to ingest. CSV file contains data that was related to the corona virus. Such as number of people caught the virus, number of deaths and number of recovered cases. JSON file holds data related to Math test results in America, from grades to 6-10. Students from different ethnicities were tested (White, Black, Asian and Hispanic). These files contained various tests, test results and the difficulty of the math test. SQL files holds data concerning students in university, ID, specialty, year, age and cycle. TEXT files contains information concerning Big Data, latest technologies and techniques concerning this domain.

b) **Classifying the files**
During this vital step, we created distinct data pipelines for each file format. We took advantage of Spark Sessions to complete the classification. For example, when working with a CSV file we created a spark session specifically for files with .csv extensions. So, any file that ended with .csv would be sent down this specific pipeline and ingested in Hadoop. You might be thinking "What if the file wasn't in CSV format?". Well with our data classifier we weren't entirely basing our classification based on the extension of the file. We are in fact entering the file and verifying line by line that indeed the format of inside the file matches the extension from the outside. To test our theory that our classifier works from inside and outside the file, we created a file that contained data in JSON format however the extension was .csv. After classifying our selected file, we were faced with numerous errors and the reason for that because the file that had JSON data inside and had a csv extension couldn't be sent down the CSV pipeline since the inside did not match the outside. This proves that our classifier enters our file, scans the data comprised in this file, compares whether the data inside matched the extension of the file and finally sends it down the correct data pipeline.

c) **Data Ingestion**

After the classification is completed, data is now being sent down specific pipelines in a neat and orderly manner. The end point of these data pipelines would also be considered their final destination and that is our Data Lake in this case we used Hadoop. Our initial data is being sent down these pipelines without any alterations happening to the actual data itself. We are guarding the raw data in AS IS format. We did apply some light transformations but it hasn't impacted the information compromised. The purpose of these transformations was to manipulate the data to send them down the data pipelines.

Not only have we established complete order while classifying our files, we've also established total organization even when loading in to Hadoop. Each file has been loaded in a nominated folder according to its formats. Example: if a csv folder after it was classified got sent down a specific pipeline, it will also be loaded in a specific segment in Hadoop.

The motivation behind organizing also the storage system was to eliminate the risk of the Data Lake slowly transforming into a swamp of data. The number one factor to as why Data Lakes turn into swamps is due to the fact that data is being loaded and stored in any random free space in the Lake. This also makes it an annoyance for the client to search and find their required data, which leads to forgotten and unused data, which brings us back to the Lake turning into a swamp.

d) **Data visualization**

Last step of this framework is to visualize the data that is now stored in Hadoop. Way before we decided to implement this feature, we had a tough time understanding our data. We placed ourselves in the clients' shoes. Clients are ordinary people; they don't have the skills nor the ability to fully comprehend what's inside the Lake. This is the reason that pushed us to add this feature in order to enhance Sparks ingestion framework. With this new touch, we managed to display our CSV, JSON, SQL and TEXT files in a well-presented manner and that is through the usage of bar graphs, line graphs, pie charts. We also applied some text classification to our TEXT files and presented the final result in tables.

6.6 Conclusion

Spark has gone through fast evolution over the past couple of years, becoming the most operative data processing and AI engine in enterprises today due to its speed, ease of use, and sophisticated analytics. This data ingestion solution seemed too good to be true. After placing it under the microscope, we found some unclarified dark spots in the framework that needed enhancement.

Data classification was the first to be noted down as an improvement. Before ingesting our data, we precisely created data pipelines to categories our data and organize it into specific bulks. On the other hand, we weren't only organizing our data before ingesting it, we were also organizing it after it was ingested. The data that was being loaded into Hadoop was likewise rationally structured according to the various data pipelines.
Data visualization was the second aspect to our contribution. After ingesting the data into our Lake, we applied a visualizer to the data in order to describe and present the data in an understandable manner to the ordinary human being.

Throughout this chapter, we clarified the various aspects we've decided to improve to Apache Sparks ingestion procedure. We thoroughly defined and explained the two added features which are: Data Classifier/Categorizer and Data Visualizer. We also provided the importance and benefits of both these elements. Then we described how we managed to adapt these features to the overall framework and how they ameliorated the ingestion procedure.

To conclude, a Data Lake must remain coherent and hold a certain value. By applying the classifier, we have organized the data into categories and preserving the data in it raw state. Moreover, the Visualizer's purpose is to tell the story of the Data Lake and stay up to date of the data inside of it.

Chapter 7

Implementation and Evaluation

7.1 Introduction

In the course of this chapter, we began be defining our new architecture after integrating our two new elements. We presented the trajectory of data that it takes [1] Extraction, [2] Classification, [3] Ingestion, [4] Storing and [5] Visualization. Next, we presented our newly implemented interface that reflects our theory. We placed it to the text and offered all the possible circumstances an end-user will face by using our interface. The screenshots provided present all the different aspects of our interface. Afterwards, we moved on to the steps we took to install, configure and integrate the two environments: Apache Spark and Hadoop. Finally, we closed this chapter with a conclusion to summarize everything in this chapter.

7.2 Architecture

In the previous chapters, we clarified all the different aspects on Apache Sparks architecture and the way it ingests data from the source and into the sink throughout its pipeline. After putting the light on the two elements that we decided that will boost the overall ingestion framework. We now would like to describe the overall new data ingestion architecture. This architecture goes numerous stages: Extraction, Classification, Ingestion and Data Visualization.We've visualized our new architecture in Figure 7.1 below.

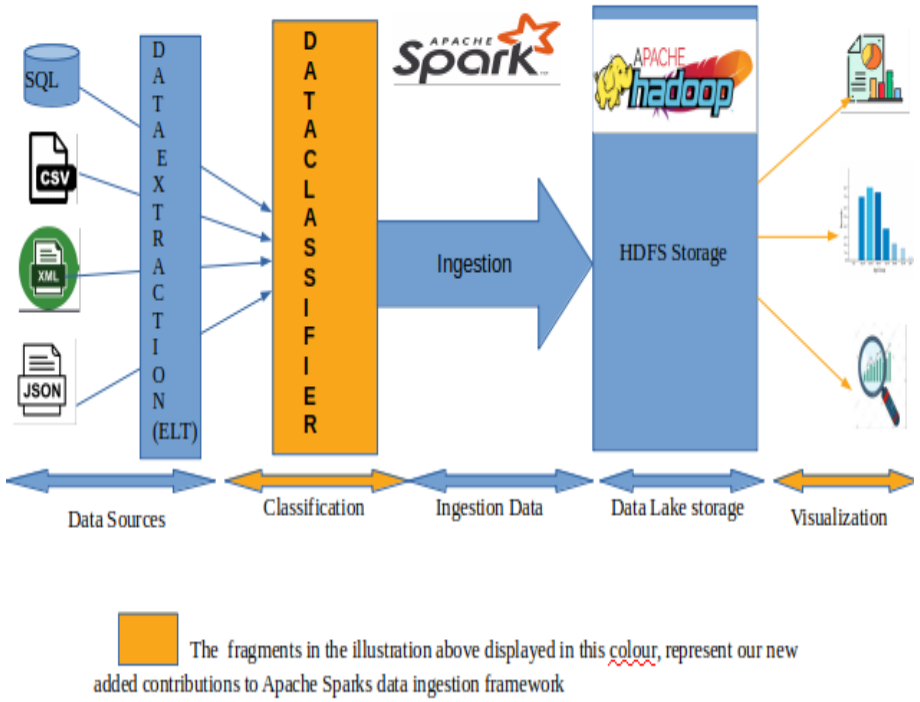

Figure 7.1: New Adapted Architecture

7.2.1 Extraction

The first step in the framework is extracting the data from various data sources. The first set of data we retrieved was from MySQL. The database called "Classic Models" holds numerous tables that concerns the number of car sales, the client's information and data that concerns the purchasing of cars. The second set of data is information that concerns number of people effected by the corona virus, number of deaths and recovered cases. This information was stored in CSV format. Third set of data is in JSON format and it is math test results in America according to the ethnicity, grade and age of students. The last data source is a knowledgably article that concerns the world of Big Data, technologies and new solutions.

7.2.2 Classification

As explained in the previous chapter, the classification procedure happens right before the ingestion phase. This procedure is essentially the creation of different pipelines according to the number of formats of files. The motivated of this segment is to classify the format of files by comparing the inside of the file whether or not it matches with the extension of the file. Right after classifying the files, the data is ready for ingestion.

7.2.3 Ingestion

This segment has been completed through the manipulation of Spark Session. After creating various data pipelines according to the format of files, we were ready to load data in to our Data Lake. As revealed previously, the technology we have selected as our Data Lake is HDFS (Hadoop Distributed File Storage). After the classification of our files was successful, we operated Apache Spark to ingest and load data into Hadoop in organized categories.

7.2.4 Visualization

To successfully and entirely append this element to the overall ingestion framework, we worked with specific libraries in PyCharm IDE to display bar and line graphs, tables and pie charts. The libraries that were used to visualize our data were: Pandas, Numpy, Matplotib, Seaborn and Codecs. These libraries were precisely created and shaped to simplify and facilitate the process to visualize data. These libraries are known to be used in Jupyter Notebook. However, we discovered that it was made possible to manipulate in PyCharm IDE.

7.3 Interface

According to Figure 7.2 is the first thing that appears when you run the app. A simple interface, that shows a toolbar from above, a text box on the left side and four progress bars on the right.

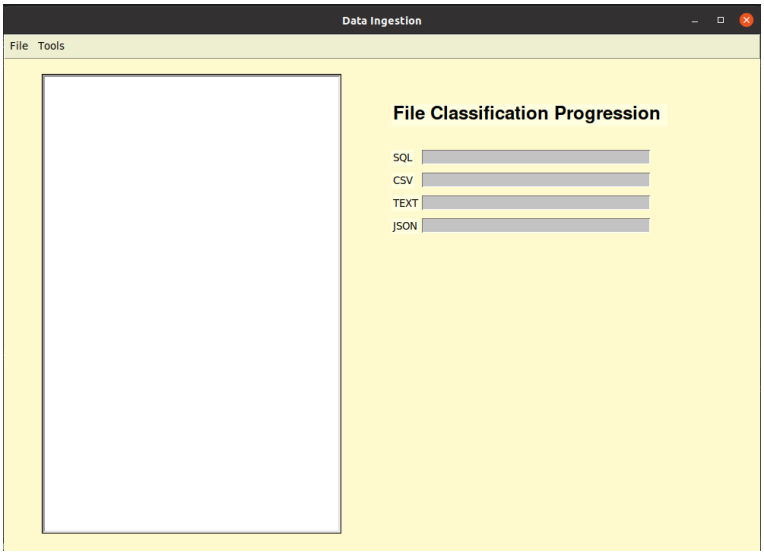

Figure 7.2: The interface

Underneath are the steps to take to manipulate the interface to get it starting.

a) First step is to extract the data. Before any action can take part, is it necessary for the client to open the File in the toolbar. In the dropdown menu, the user will be face with two elements: Available Formats and Folder Location. Taking a look at Figure 7.3, is an image displaying the tool bar in our interface.

   i. Available Formats: This element displays the file formats this framework is capable of ingesting. This component was specifically created to inform the end user that the following file formats are possible to ingest.

   ii. Folder Location: This segment allows the end user to select numerous Folders that they wish to ingest. We created this component in the aim to provide freedom to the user to freely select folders all over their machine. We could've facilitated our job by determine a specific folder to place their data and limit their selection. However, we gave users the ultimate freedom to select folders no matter where they're found.

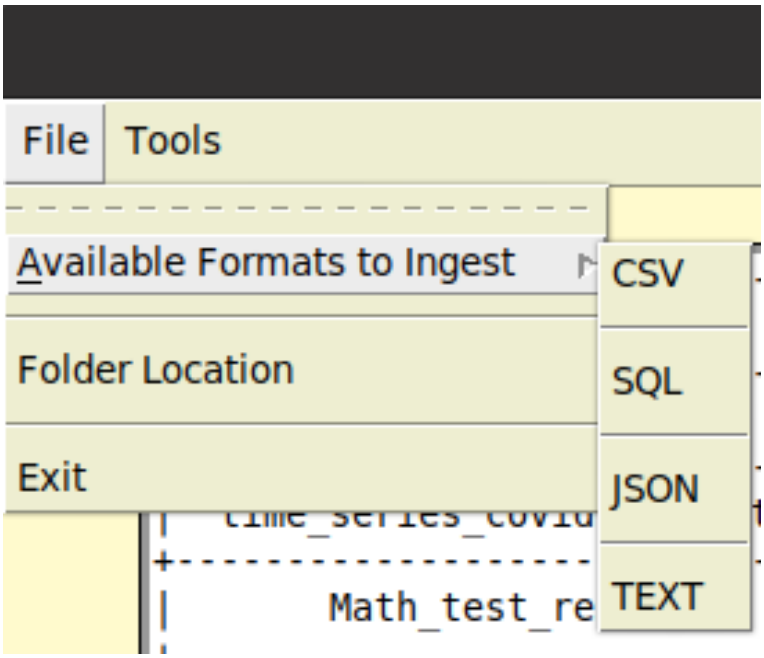

Figure 7.3: Menu Bar

After the user has clicked on “Folder Location”, a pop up appears, that allows them to select a specific directory.You can clearly notice the pop up in Figure 7.4. The files that are found in the directory or folder that’ve been selected, are the same files that will be ingested later on.

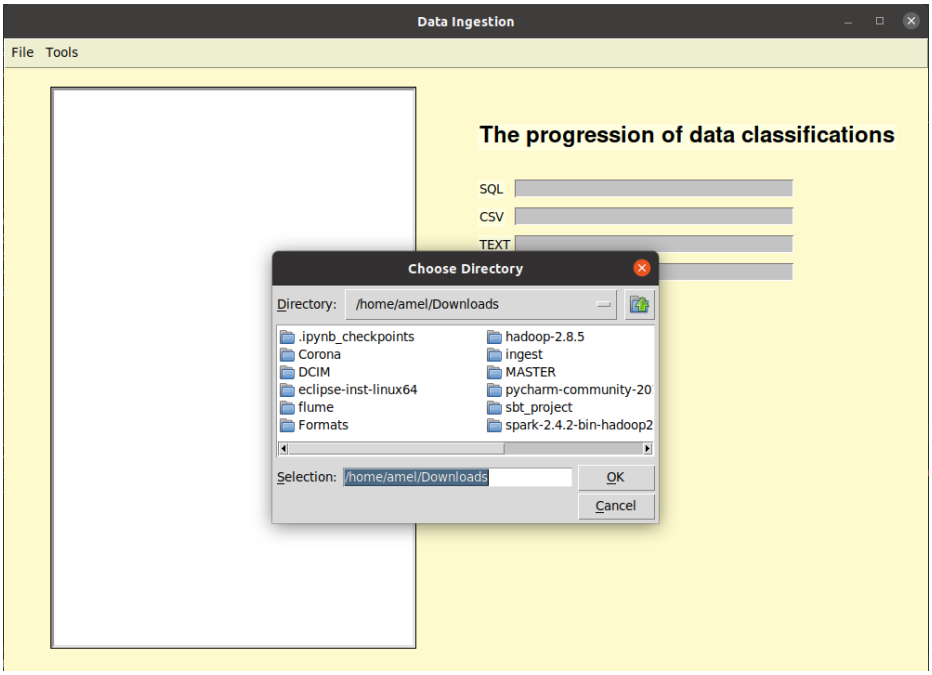

Figure 7.4: Extract Sources

When the user has decided which folder to use and has completed the selection. In Figure 7.5 as displayed in the textbox, a table will appear that holds two columns.

The first column displays the name of the file and the second column displays the format of the file.

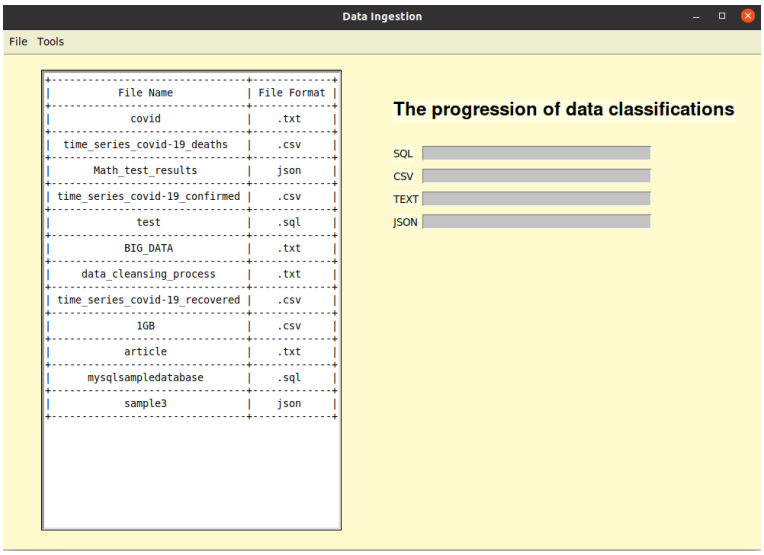

Figure 7.5: Selected Folders

We displayed the contents of the selected folders in a text box in the aim of providing the user with the files and the formats they desire to ingest. This presentation is another way to allow the client to confirm these are the files they want to load for the second and last time.

b) Second step is where the role of file classification makes an appearance. If you glance over to Figure 7.6, the user will come across the second segment of the toolbar which is “Tools”. Here the user can decide how they would like to operate the data.

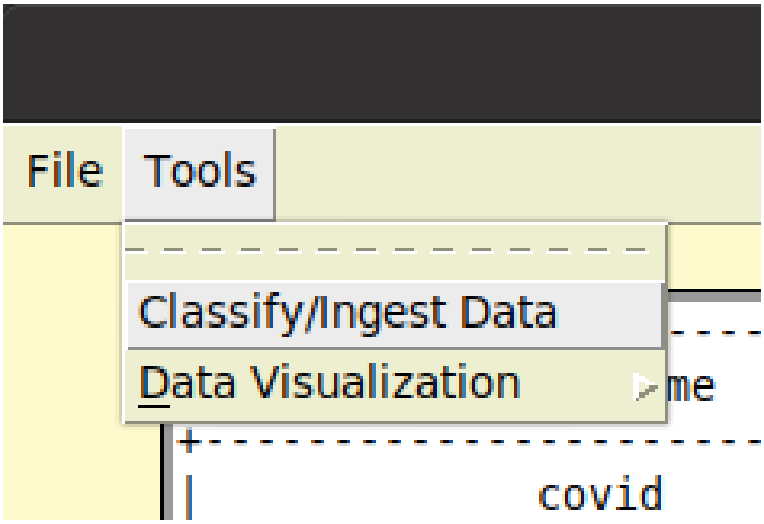

Figure 7.6: Classification tool bar

This is where our contribution to the ingestion framework is presented. After the client has selected their anticipated folder, the source code we implemented begins by classifying the files according to their extension. After the files has now places in designated categories, then the classifier enters the file and proceeds to read through the data line by line to confirm that in fact the file is in the correct format.
During the classification, you can notice the progress bars below advancing. This is displaying the progress of the classification that is happening behind the curtains. As you can see, the categorization is completed format after format. This means the files are being sent all at once and pass-through check points. Then each file falls into its designated category. The question we would like to make clear is "Why didn't the classification of formats work in parallelism?". We know Spark is popular for its work in parallelism, we wanted to implement this feature in to our work, but due to the fact that our machines were not efficient enough to handle the stress.

So instead, we worked our classifier, format by format and classified them in a non-paralleled manner. The image below displays how the advancement of each classifier occurred.
Figure 7.7, displays the advancement of our data pipeline while they're in full swing.

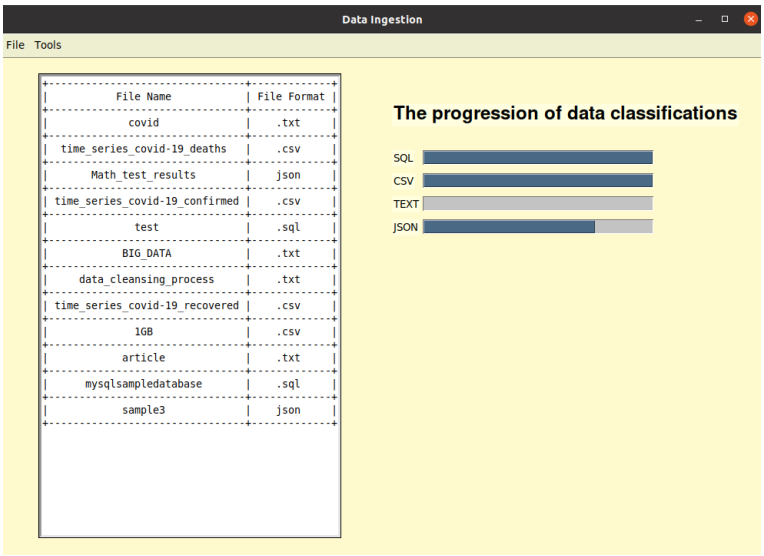

Figure 7.7: Classification Progress

The message box that appears and is displayed in Figure 7.8 underneath is designed to inform the user that the categorization is accomplished and that the files have been effortlessly parsed through a classifier and have been placed in nominated categories.

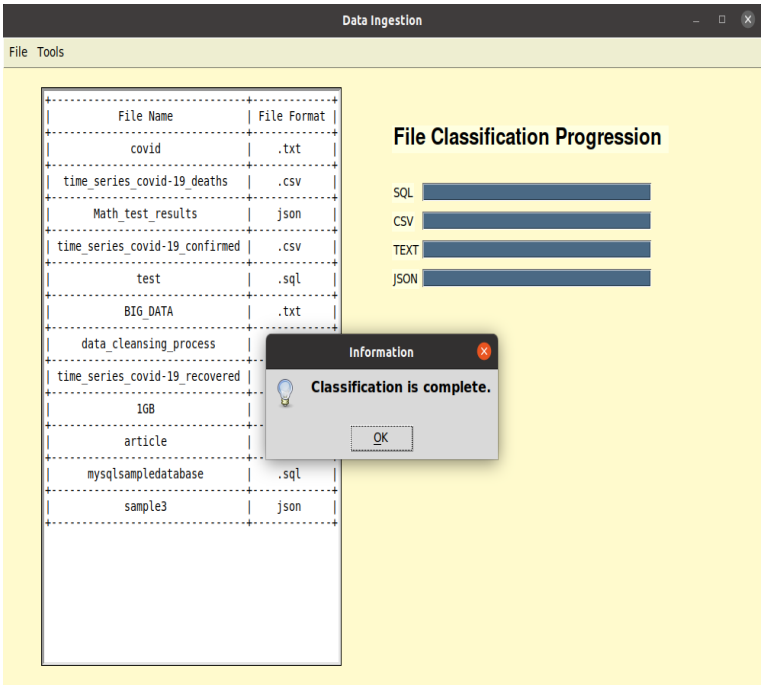

Figure 7.8: Message Box:Classifications complete

c) The third stage of this framework is: Data Ingestion. After neatly organizing our files into rationally categorized segments in the aim of gaining order of the system, it is now time to ingest. Before beginning any type of ingestion, a message box appears to notify the user that they're establishing connection to Hadoop to commence loading the data.

Figure 7.9 underneaths shows the message box

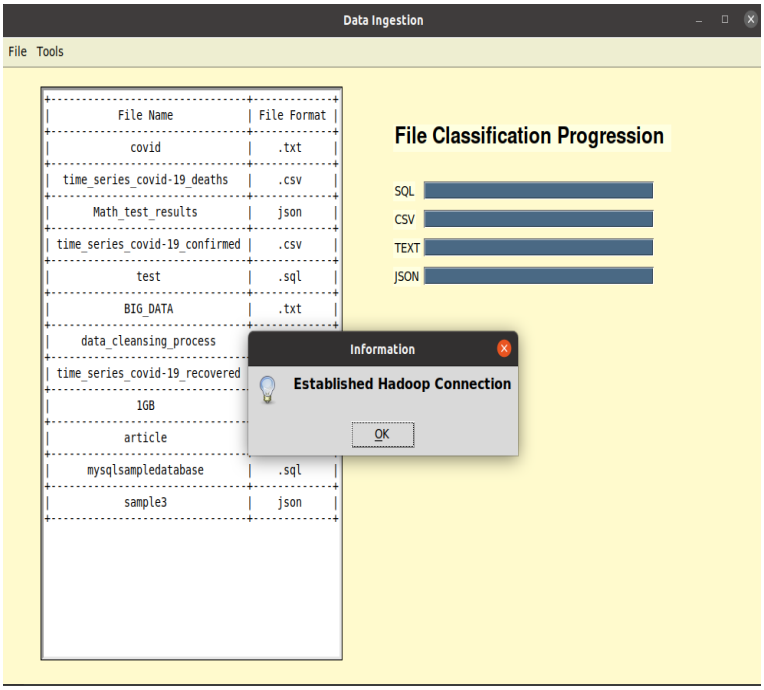

Figure 7.9: Message Box:Connected to Hadoop

On top of Spark's framework, as mentioned earlier, we generated numerous data pipelines according to the number of formats. These various pipelines continue the flow of data and transport them into Hadoop. We maintain our goal to keep the Data Lake organized and we do so by ingesting the files according to their formats. Each format has a specific folder in Hadoop. This signifies that the order and organization is kept on top of the game in the DL. Example: All CSV, SQL, TEXT and JSON files were stored in their folders named CSV, SQL, TEXT and JSON.
After the ingestion is completed, a message box that is showed in Figure 7.10 underneath is a way to inform the client that the ingestion is completed.

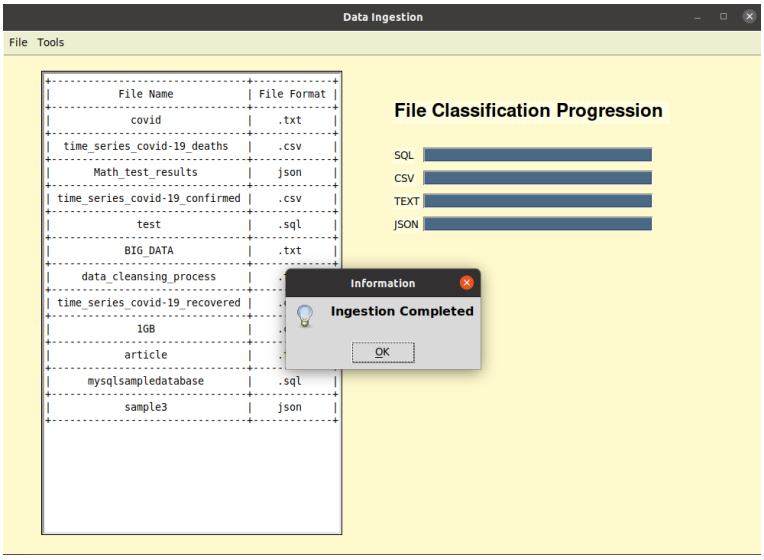

Figure 7.10: Ingestion complete

As explained before, the files are stored in the Data Lake according to the format of the files. Figure 7.11, is a representation of the Lake in Hadoop to present how the files were ingested. This is our strategic idea to ingest and load the data and keep the Data Lake organized and remove any potential risk of it transforming into a swamp.

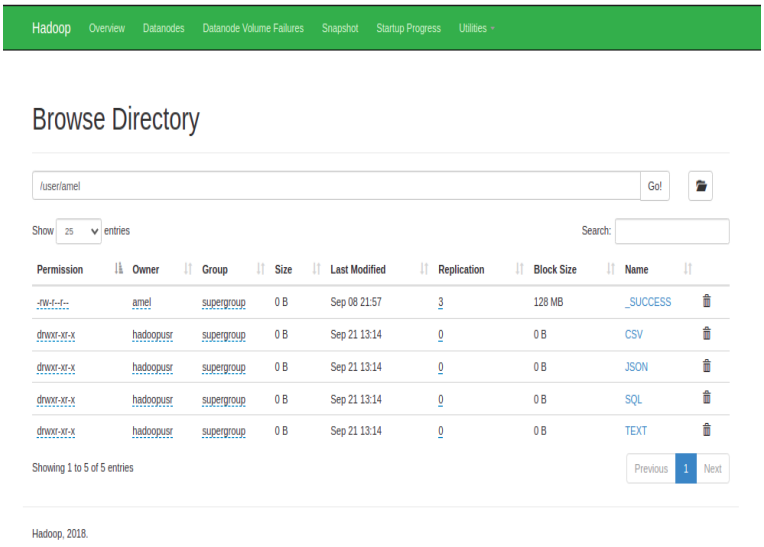

Figure 7.11: Data Lake in Hadoop

By organizing our Data Lake in to explicit categories, we have also simplified the users experience when exploring the DL to extract their required information. It has become less time consuming and less efforts required to do so.

d) The fourth and last stage of this framework: Data Visualization. If the user has a slight doubt of the data within the files, they have the choice to visualize and display in a manner that is more descriptive a comprehendible.Figure 7.12 underneath, displays where the user must select in order to visualize their data.

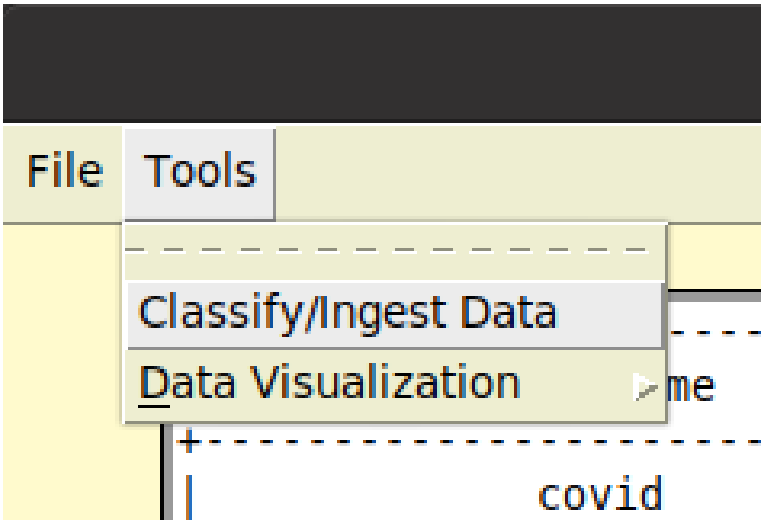

Figure 7.12: Visualization Tool Bar

Underneath are the graphs and pie charts that describe the contents of a couple of csv files. As displayed in figure 7.13 below, the csv files hold data that concerns the number of infected people by the corona virus, the number of death rates and the number of recovered cases. The graph below illustrates the number of mortality rate of the corona virus over a period of time.

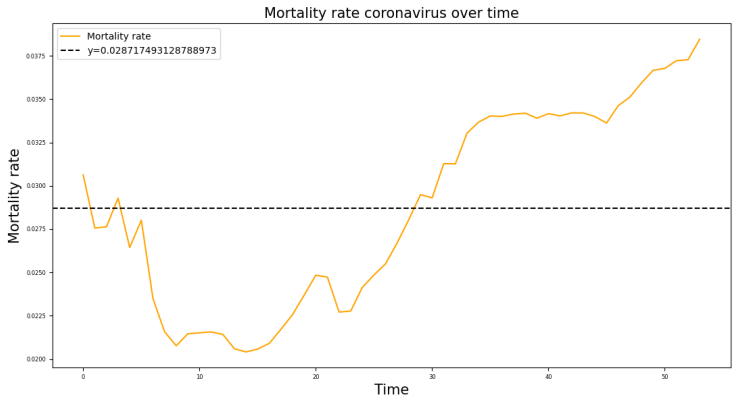

Figure 7.13: Number of mortality rate of the corona virus over a period of time

The bar graph and pie chart in figure 7.14 both display the number of Corona confirmed cases around the world.

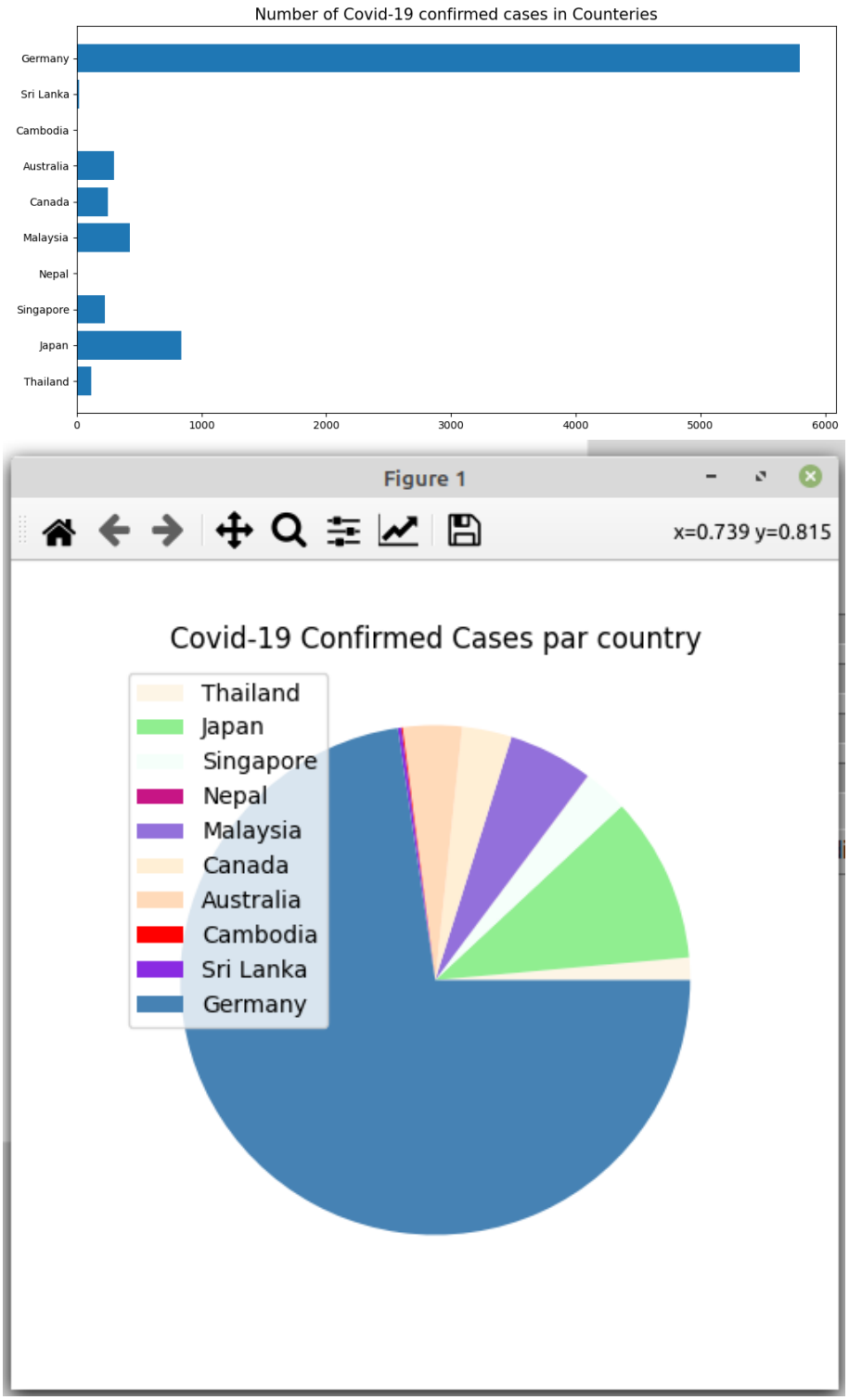

Figure 7.14: Number of Corona confirmed cases around the world

In addition, the graph in Figure 7.15 is to visualize the data in a JSON file. The data below represents the number of students tested in a math test according to their ethnicity (Asian, Black, Hispanic and White).

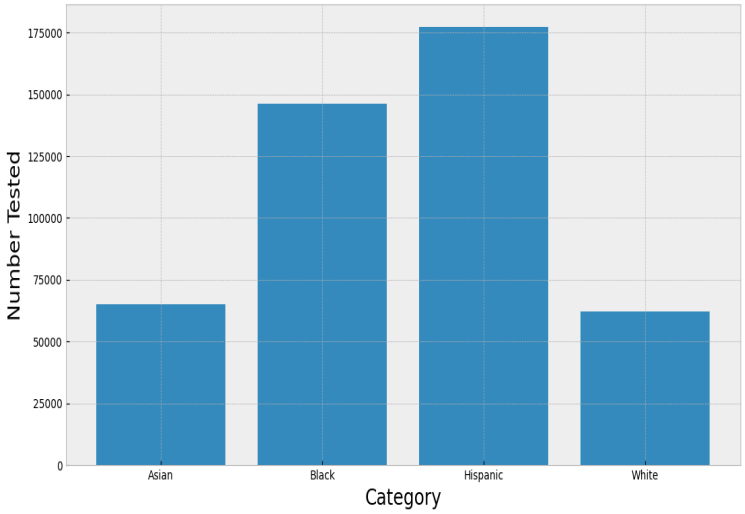

Figure 7.15: Number of students tested in a math test

Another example of our data visualization. Underneath is Figure 7.16 and it is a description of our SQL file. The data presented below describes the average salaries of employees dating from 2003 to 2005.

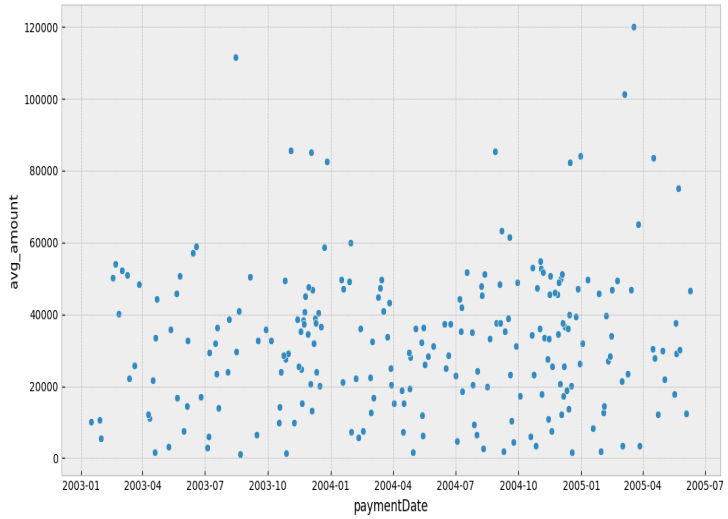

Figure 7.16: The average salaries of employees

7.4 Implementation tools

The main characters in this ingestion scenario were Apache Spark and Hadoop. These two environments are known to be massive, hefty and difficult to install, hence they both have numerous configurations to be completed and a whole different story to integrate the two together. Underneath are the following steps we took to accomplish this tough challenge.

Step1:Download java8 In ubuntu terminal:
Figure 7.17 & Figure 7.18 is what was showed in ubuntu terminal to check and complete java installation:

a) Update the package index: Sudo apt update

b) To install java8: Sudo apt install oracle-java8-installer

c) Check the java version: Java -version

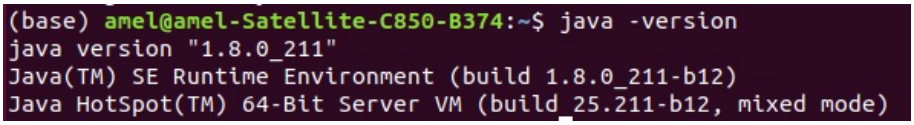

Figure 7.17: Java version

d) Install the JDK: Sudo apt install default-jdk

e) To check: javac -version

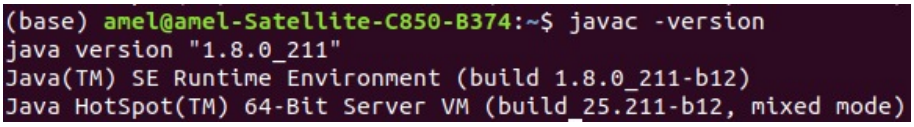

Figure 7.18: Javac version

Step2:Add user for Hadoop environment

a) Add a Hadoop system user using below command:

b) Sudo addgroup Hadoop

c) Sudo adduser –ingroup hadoop hadoopusr

Step3:Configure SSH

a) sudo apt-get install openssh-server openssh-client

b) ssh-keygen -t rsa

c) cat /.ssh/id-rsa.pub » /.ssh/authorized-keys

d) To verify the installation of ssh: ssh localhost

Step4: Download Hadoop

a) Visit the Apache Hadoop website and download hadoop2.8.5 version.

b) Go to where Hadoop was installed and run this command to extract the file:Sudo tar xzf Hadoop-2.8.5.tar.gz

c) Rename hadoop-2.8.5 as Hadoop: Sudo mv hadoop2.8.5 hadoop

d) Give the privilege: Sudo chown -R hadoopusr:Hadoop Hadoop

Figure 7.19 is the result we received after verifying that HDFS was successfully installed alongside of it the version.

```
(base) amel@amel-Satellite-C850-B374:~$ hadoop version
Hadoop 2.8.5
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 0b8464d75227fcee2c6e7f2410377b3d53d3d5f8
Compiled by jdu on 2018-09-10T03:32Z
Compiled with protoc 2.5.0
From source with checksum 9942ca5c745417c14e318835f420733
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.8.5.jar
```

Figure 7.19: Hadoop version

Step5: Configure Hadoop

Figure 7.20, are all the paths we've added to the /.bashrc file

a) Add all the paths /.bashrc file

```
#JAVA HOME directory setup
export JAVA_HOME=/usr/lib/java/jdk1.8.0_211
export SBT_HOME=/usr/share/sbt/bin/sbt-launch.jar
export SPARK_HOME=/home/amel/spark
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$SBT_HOME/bin
export PATH=$PATH:$SPARK_HOME/sbin
export PATH=$PATH:$SPARK_HOME/bin
export SCALA_HOME=/home/amel/Downloads/scala-2.12.8.deb
#HADOOP VARIABLES
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/home/amel/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/amel/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/home/amel/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/amel/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

Figure 7.20: Bashrc file

Figure 7.21, Figure 7.22, Figure 7.23 presents the configuration of the Name node, Data Node, MapReduce and Hadoop tmp files.

b) Enter /usr/local/Hadoop/ and modify all the xml files

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
 </property>

<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///usr/local/hadoop_store/hdfs/namenode</value>
 </property>

<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:///usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Figure 7.21: Configure Namenode and Datanode

```
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
</property>

<property>
<name>mapreduce.framwork.name</name>
<value>yarn</value>
</property>
</configuration>
```

Figure 7.22: Configure MapReduce

```
<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
 </property>
</configuration>
```

Figure 7.23: Configure Hadoop tm

c) $ HADOOP-HOME/bin/hdfs name node -format

d) Start Hadoop single node cluster using below command: Start-all.sh

e) To confirm everything is working: Jps
Figure 7.24 is a command to use in ubuntu to versify that Hadoop is running correctly.

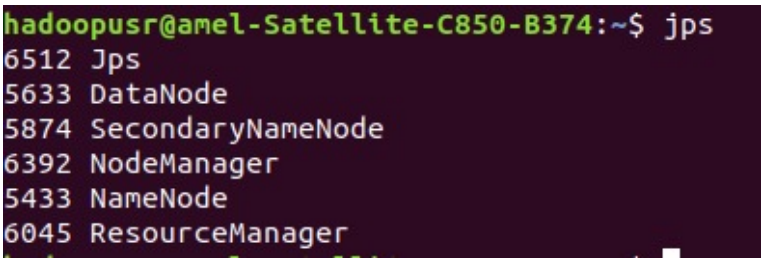

Figure 7.24: Jps command

f) Figure 7.25 is the page that appears after typing “Localhost50070” on a navigator in order to confirm Hadoop has been successfully downloaded, configurated and is operating

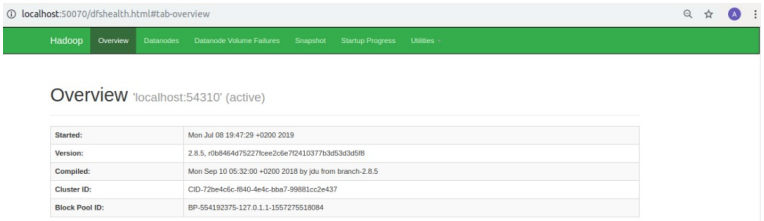

Figure 7.25: HADOOP on Navigator

g) Stopping Hadoop: stop-all.sh

Step8:Install Spark

a) Sudo apt-get install git

b) Next, go to https://spark.apache.org/downloads.html and download a pre-built for Hadoop 2.7 version

c) Extract the file: Tar xvf spark-2.0.2-bin-hadoop2.7.tgz

d) Cd spark-2.0.2-bin-hadoop2.7.tgz

e) Cd bin

f) spark-submit –version
Figure 7.26 displays that spark is installed and is operating correctly.

Figure 7.26: Spark version

g) Figure 7.27 is a window that appears when typing on a navigator “localhost4040’ in order to confirm Spark has been successfully downloaded and configurated:

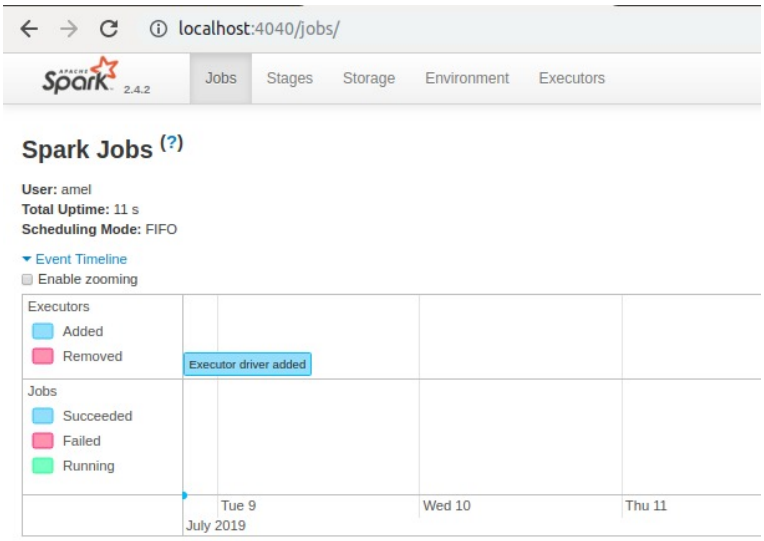

Figure 7.27: Spark on navigator

7.5 Conclusion

Throughout this chapter, our aim was to shine light on our contributions that were developed to enhance the data ingestion procedure. We started off by explaining our new data ingestion architecture and clarified where and how our new added elements were integrated in the framework. We described all the steps included in the new architect [1] Data Extraction, [2] Data Classification, [3] Data Ingestion and [4] Data Visualization. We then proceeded to explain our freshly created interface in order to bring life to our theory and integrate the Categorizer and Visualizer. We deeply explained the steps an end-user would go through and the results they will witness. Lastly, we illuminated the steps we took to install, configure and integrate the large framework, both Apache Spark and Hadoop. We mentioned only these two since they were large, time consuming and extremely difficult to setup.

Chapter 8

General Conclusion

8.1 Conclusion

This thesis report is aimed to treat a defined segment from Data Lake and that is the Data Ingestion of various formats of data. Since data arrives in to a DL in different varieties, unorganized and complicated to comprehend. It was our main objective to integrate new approaches to enhance the overall procedure.

The first segment of this paper contains all the fundamental information that concerns this topic. We started of by chapter one which contains information about another data storing system known as Data Warehouse. We began by explaining how DW operates. How it extracts, cleanses, transforms then loads data. We then moved on to describe the different type of existing warehouses, different schemas that are used and its overall architecture. We deeply defined how OLAP and OLTP are used and manipulated in a DW. Finally, listed all the benefits and inconveniences of using this type of data storage.
In chapter two, we explained what makes data so big (Volume, Variety, Veracity, Velocity and Value). We later on listed the challenges and benefits that follow big data. Lastly, a detailed illumination on the Big Data's ecosystem. Throughout chapter three, we went in depth of the Data Lake storage system. We listed the importance, characteristics, extracting value and the philosophy of a DL. Next, we clarified the different technologies that are used in a Lake such as Storage, Data Source, Data Ingestion, Data profiling, Processing and Data governance. We described both ETL and ELT processes and provided the differences amongst the two. To close this chapter, we compared both Data Lake and Warehouse and listed their differences.
The last chapter of this segment is Data Ingestion. Since we know there are various components of DL, we made it clear that we will be zooming into the overall Data Lake architecture and working on the Data Ingestion segment only. All through this chapter, we systematically managed to illuminate the architecture of the ingestion process. We

placed down all the layers included (Data Collector, Processing, Storage, Query and Visualization). We jotted down the different parameters of the ingestion operation (Data Lineage, Velocity and Frequency). A broad definition of data pipelines was included, also the CDC (Change Data Capture) procedure. To finish off this chapter, two popular tools were mentioned: Apache Sqoop and Kafka.

The second segment holds all the related work related to our thesis project. Throughout chapter five we provided our research on plenty data ingestion solutions: Apache Spark, Apache Gobblin, Marmaray and Lambda. Each and every single solution includes and explanation of its architecture and the different operations it takes to completed their data ingestion. We provided the challenges of each data ingestion solutions we managed to find and their distinct data pipelines to ingest data. We then proceeded to compare the solutions and placed those 92comparisons in a structured table. In the conclusion, we made it clear that we've adopted Apache Spark to integrate our new contributions and ameliorate the overall ingestion framework.

The last segment of our report is the contributions fragment. In the course of this chapter, we made clear all the elements we've came to conclusion that will enhance and boot Spark's data ingestion framework. After methodical research, we found out the framework was missing essential components. We provided a strategical explanation on how our two added modules (data categorization & visualizer) will improve and enhance the ingestion operation after being integrated in to Apache Sparks architecture. We then put into words and well-defined what is a data classifier and visualizer, their importance and benefits. Next, we explained how attaching these two modules will impact Sparks ingestion framework in total. Following, we provided multiple screenshots concerning our developed interface. We described all the different scenarios a user can confront during his interface experience. We went to full length to describe the following steps; Extraction, Classification, Ingestion, Storing and Visualization. Each and every single step was attached a screenshot of the interface to help understand the overall concept.

To answer the question why we chose to work on this topic. Out of all the environments, we chose to work on the ingestion of heterogeneous data in a Data Lake environment. Data Lakes have essentially become the base of every large company, just because they offer simpler and flexible options to scale when working with massive and varied data. The other reason to as why we chose the data ingestion fragment is because, many companies have dealt with many issues from data being generated from various data sources. Here data ingestion main role is to consolidate data and storing it in the Lake. By using the correct data ingestion tools, major organizations can extract, load and transform data for personal

use. Data Lakes were the proposed solutions regarding the limitations that came with Data Warehouses. Since they use ELT tools, the processing of data is less time consuming when comparing to Data Warehouse. In addition, the number one challenge when it comes working with Data Lakes, is extracting and slightly prepossessing the data before ingesting it into the Lake. Scientist have complained that it's one of the major challenging aspects when working in a Data Lake.

After examining all the possible solutions of data ingestion, we came down to the four most used approaches which are: Apache Spark, Apache Gobblin, Marmaray and Lambda. The data ingestion approaches all possess the ability to ingest large data that is arriving from multiple data source at different velocity. However, they all share the same inconveniences. The cons of these approaches is that they all ingest and load data in a messy manner. If helplessly continuing to use these approaches that have horrible organization methods, the Data Lake is at risk of it turning into a swamp, making it more time consuming when locating data, losing its complete value and integrity and just overall making it a horrible experience for users.

In the frame of our thesis project, we strongly believe that our contribution has ameliorated the overall ingestion operation of various data formats. Our new appended approach has developed Apache Spark ingestion procedure in general. It is true, Spark does have the ability to ingest data. However, it performs the ingestion in an unorganized and chaotic manner. As brought up earlier, if using Sparks framework as it is, the Data Lake has a high risk of it turning into an ocean or a swamp of data. We implemented a data categorizor before ingesting the data. To simply put in words, the classifier works on organising the data before its sent down to be ingested and also organises the actual Data Lake itself. To summarise, we've reduced the risk of the Lake turning to a swamp of data, made it easier to locate data. Meaning now its less time consuming when extracting data. Guarded the value and integrity of the Lake and just overall enhanced the users experience when using it. Another element we added to our new approach is the data visualizer. This element helps the end user to visualize data, understand the story, pick up on trends and new insights, take better decisions, analyse the data and detect any errors. So when placing the old and new approach together, it is without doubt that our new approach has solved the limitations that were discovered in the old approach.

8.2 Future Work

Concerning our work, it is no doubt that the framework has a lot more developing to be integrate in to, just like any other work. The work that we achieved was to treat a small aspect in the overall Data Lake ecosystem and that is Data Ingestion. We zoomed into the ingestion component and treated an aspect of it an that is the classification and categorization of data. we do believe that our work still has a lot of improvements that can be applied to, and the list below are the future work we plan to integrate.

- Parallelization:
When taking a look in the third section of this report and reading deeply. You can see that our data classifiers were created in an unapparelled manner. Even though we are working on Apache Spark environment which is generally popular for its parallelization. The reason why we didn't implement this aspect was due to the reason that our machines weren't sufficient enough to handle Spark working all the data pipeline in parallel of each other. By implement this aspect in the future, it will ameliorate the overall ingestions framework by excelling it into becoming less time consuming. Working all the data pipeline will save us a lot more time compared to launching the pipeline one by one, then continuing to ingest the categorized data one by one also.

- Big Data Semantics:
A quick definition of Data Semantics, it refers to the "meaning and meaningful use of data". For our future work we plan to achieve implementing this element in to our new data ingestion solutions. An example how this component would be implemented; for instance, a user would like to write an article that concerns "Natural Disasters" and would like to extract any information that is linked to that topic in their Data Lake. How we plan to implement it is the user begins to type in a couple of keyworks related to their required research, then we proceed to collect all the different files that are in relation of those keywords. Whether the data was found in text, csv, photos, videos, json files. All of them would be collected and presented to the user to complete their research. We plan to create relationships or gateways between all the data that fundamentally exists in the Data Lake. Since the storage system of a Lake is immensely vast, searching for data will become a tough challenge for users. By integrating this in the future, it will help in searching the Lake, corner by corner and extracting data. Creating relationships between data will help unify the Data Lake.
We do wish to implement this feature after the data has been ingested into the Lake. Big data semantics will be integrated and applied to the Lake itself, in order to extract all the data that is related to a certain topic.

-Metadata:

Metadata is data that describes other data. In information technology, the prefix meta means “an underlying definition or description.” So, metadata describes whatever piece of data it’s connected to whether that data is video, a photograph, web pages, content or spreadsheets. To realize maximum value from our data lake, we must be able to ensure data quality and reliability, and democratize access to data. We aim making it faster for users to identify the data they want to use. All of this critical functionality is dependent on putting in place a robust, scalable framework that captures and manages metadata. Metadata is truly the key to a successful next-generation data architecture.

Bibliography

[1] Achari, S. (2015). *Hadoop essentials*. Packt Publishing Ltd.

[2] Adamson, C. (2012). *Mastering data warehouse aggregates: solutions for star schema performance*. John Wiley & Sons.

[3] Ahmad, A. K., Jafar, A., and Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*, 6(1):1–24.

[4] Alwidian, J., Rahman, S. A., Gnaim, M., and Al-Taharwah, F. (2020). Big data ingestion and preparation tools. *Modern Applied Science*, 14(9).

[5] Bala Mahfoud, B. O. and Zaia, A. (2016). Extracting-transforming-loading modeling approach for big data analytics. *International Journal of Decision Support System Technology (IJDSST)*, 8(4):50–69.

[6] Bhardwaj, A., Kumar, A., Narayan, Y., Kumar, P., et al. (2015). Big data emerging technologies: A casestudy with analyzing twitter data using apache hive. In *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*, pages 1–6. IEEE.

[7] Buenrostro, I., Tiwari, A., Rajamani, V., Pattuk, E., and Chen, Z. (2018). Single-setup privacy enforcement for heterogeneous data ecosystems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1943–1946.

[8] Gupta, S. and Giri, V. (2018a). Data lake ingestion strategies. In *Practical Enterprise Data Lake Insights*, pages 33–85. Springer.

[9] Gupta, S. and Giri, V. (2018b). *Practical Enterprise Data Lake Insights: Handle Data-Driven Challenges in an Enterprise Big Data Lake*. Apress.

[10] Imhoff, C., Galemmo, N., and Geiger, J. G. (2003). *Mastering data warehouse design: relational and dimensional techniques*. John Wiley & Sons.

[11] John, T. and Misra, P. (2017). *Data lake for enterprises*. Packt Publishing Ltd.

[12] Kannan Sobti, P. and Dash, D. (2020). Top big data technologies for data ingestion.

[13] Kimball, R. and Ross, M. (2011). *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons.

[14] Kune, R., Konugurthi, P. K., Agarwal, A., Chillarige, R. R., and Buyya, R. (2016). The anatomy of big data computing. *Software: Practice and Experience*, 46(1):79–105.

[15] Laptev, N., Smyl, S., and Shanmugam, S. (2017). Engineering extreme event forecasting at uber with recurrent neural networks. *UBER Engineering*, 9:06–17.

[16] Liu, R., Isah, H., and Zulkernine, F. (2020). A big data lake for multilevel streaming analytics. In *2020 1st International Conference on Big Data Analytics and Practices (IBDAP)*, pages 1–6. IEEE.

[17] Mahfoud, B., Omar, B., and Zaia, A. (2017). A fine-grained distribution approach for etl processes in big data environments. *Data & Knowledge Engineering*, 111:114–136.

[18] Mathis, C. (2017). Data lakes. *Datenbank-Spektrum*, 17(3):289–293.

[19] Meehan, J., Aslantas, C., Zdonik, S., Tatbul, N., and Du, J. (2017). Data ingestion for the connected world. In *CIDR*.

[20] Munshi, A. A. and Mohamed, Y. A.-R. I. (2018). Data lake lambda architecture for smart grids big data analytics. *IEEE Access*, 6:40463–40471.

[21] Orozco-GómezSerrano, A. (2020). Adaptive big data pipeline.

[22] Perrin, J. (2020). *Spark in Action, Second Edition: Covers Apache Spark 3 with Examples in Java, Python, and Scala*. Manning Publications.

[23] Qiao, L., Li, Y., Takiar, S., Liu, Z., Veeramreddy, N., Tu, M., Dai, Y., Buenrostro, I., Surlaker, K., Das, S., et al. (2015). Gobblin: Unifying data ingestion for hadoop. *Proceedings of the VLDB Endowment*, 8(12):1764–1769.

[24] Room, C. (2020). Exploratory data analysis. *machine learning*, 8(39):23.

[25] Sawant, N. and Shah, H. (2013a). Big data application architecture. In *Big data Application Architecture Q & A*, pages 9–28. Springer.

[26] Sawant, N. and Shah, H. (2013b). Big data ingestion and streaming patterns. In *Big Data Application Architecture Q & A*, pages 29–42. Springer.

[27] Smart, M. (2021). *Learn Excel 365 Expert Skills with the Smart Method: Fifth Edition: Updated for the Jan 2021 Semi-Annual Version 2008*. Smart Method Limited.