

QUALIFYING PAPER

For the degree of

Master

(degree name)

topic: Data Quality Management in ETL Process under Resource Constraints

Submitted by: six year student _____, group ICAM-62

specialty 124 System analysis

(code and name of specialty)

(signature)

A. Kashosi

(surname and initials)

Supervisor

(signature)

N. Zagordna

(surname and initials)

Standards verified by

(signature)

O. Matsiuk

(surname and initials)

Head of Department

(signature)

I. Bodnarchuk

(surname and initials)

Reviewer

(signature)

(surname and initials)

Ministry of Education and Science of Ukraine
Ternopil Ivan Puluj National Technical University

Faculty _____
(full name of faculty)

Department _____
(full name of department)

APPROVED BY
Head of Department

(signature) I. Bodnarchuk
(surname and initials)
« » 2022

ASSIGNMENT
for QUALIFYING PAPER

for the degree of _____ Master _____
(degree name)

specialty 124 System Analysis _____
(code and name of the specialty)

student _____ Kashosi Aser _____
(surname, name, patronymic)

1. Paper topic Data Quality Management in ETL Process under Resource Constraints

Paper supervisor Nataliya Zagorodna, PhD, Associate Professor _____
(surname, name, patronymic, scientific degree, academic rank)

Approved by university order as of « 22 » 11 2022 № 4/7-951

2. Student's paper submission deadline _____

3. Initial data for the paper _____

4. Paper contents (list of issues to be developed)

5. List of graphic material (with exact number of required drawings, slides)

ANNOTATION

Data Quality Management in ETL Process under Resource Constraints // Qualification paper of the educational level "Master" // Kashosi Aser // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science // Ternopil, 2022 // P. , Tables – , Fig. – , Diagrams – , Annexes. – , References – .

Keywords: Data Quality, ETL, Big Data, Data Management Platform, Stratified Sampling

Currently, access to data is necessary for many companies, particularly those engaged in marketing, to make decisions that will improve the quality of their services and businesses. They frequently find the knowledge they need from several sources in a variety of formats. Following a dedication to the quality of information offered to data consumers, a system will be implemented to consolidate all these data sources for analysis and decision-making.

This study addresses the evaluation of data quality (DQ) in an ETL process developed to support a marketing data management platform. More specifically, this study addressed the problem of evaluating the quality of data with a high-volume trait. Addressing the problem of DQ assessment at high ingestion rates is beyond the scope of this study, which focuses on data quality assessment with limitations to vertical or horizontal scaling of the ingestion system.

We also analyze the use of the model developed on real data to assume an improvement in the quality of the data in the ETL. The methodology used consisted of studying each feature related to the characterization of high-quality data and analyzing the impact of those in an ETL concerned with voluminous data. We propose algorithms for improving a more generalizable integration DQ assessment. We conducted a practical implementation study of the different criteria and characteristics proposed to evaluate the impact of the data collected throughout the process of data Extraction, Transformation, and Loading.

We highlight a quality assessment framework that models the different necessary parts of the process, including data sources, metrics characterizing data quality, data destination, and the analysis and performance of the algorithms used in the assessment process. The ETL practical implementation in this research is based on a Direct Acyclic Graph (DAG) model, with the main purpose of extracting, transforming, and transmitting data from this first service to the rest of the Marketing Data Management Platform infrastructure, which is considered as the end user. The evaluation and quality are based on the development of algorithms that take source data as input in combination with predefined properties encompassing the expected result of the ETL transformation to produce the evaluation result.

The evaluation findings may be used to support or contradict the standards for quality. Decisions are made in the event of a DQ failure to improve and enhance the data. We suggest including data checks at the very end of the ETL data manipulation process as well as a model for data volume reduction using algorithms that are intended to make the procedure more generic to enable quick review. The quality of the data evaluated during the test is a statistical representation of the ingested dataset, which provides an accurate profile that enables user applications to retrieve high-quality data without delay. The main contributions of this thesis are: i) the development of an ETL service in a Marketing Data Management Platform and ii) an examination of data reduction models with a view to assessing data quality.

Chapter 1 presents a literature review of this research and describes the basic concepts and their definitions in other research, including sampling, ETL, Data Quality and Big Data.

Chapter 2 We present the manner in which the ETL system fits into the framework of the data-management platform and how the entire architecture is modelled.

Chapter 3 presents the outcomes of the experiment. The experimental findings, which were obtained using various types of actual data, are presented in this chapter. The performance over time and the effect of the stratified sample are depicted in graphs.

The closing part presents the conclusions of this thesis and discusses the prospective research.

LIST OF SYMBOLS, SYMBOLS, UNITS, ABBREVIATIONS AND TERMS

DQ – Data quality

ETL – Extract transform and load

DMP – Data management platform

DAG – Directed acyclic graphs

BD – Big data

TPC – Transaction Processing Performance Council

DI – Data integration

ISO – International Organization for Standardization

IEC – International Electrotechnical Commission

BSP – Bulk Synchronous Parallel

SVD – Singular value decomposition

PCA – Principal components analysis

DB – Database

CPU – Central processing unit

CONTENT

Data Quality Management in ETL Process under Resource Constraints	1
Notes	3
INTRODUCTION.....	9
1.1 ETL	12
1.1.1 ETL definitions	13
1.1.2 ETL tools review	14
1.2 Data quality	16
1.2.1 Data Quality Dimensions.....	17
1.2.2 Data Quality Objectives in the Context of ETL	18
1.2.3 ISO Data Quality Standards	19
1.3 TcP-di benchmark	20
1.4 Big data	21
1.4.1 Vs and BIG Data.....	21
1.4.2 Batch processing and Big Data.....	22
1.5 Sampling for big data.....	23
1.5.1 A taxonomy for Big Data sampling techniques	24
CHAPTER 2. SYSTEM MODELING	26
2.1 ETL model	27
2.2 System architecture overview	29
2.2.1 Metadata store.....	30
2.2.2 Horizontal autoscaling environment.....	31
2.2.3 Workflow runner	33
CHAPTER 3. MEASUREMENT RESULTS.....	35
3.1 Estimation of the Population Mean	36
3.2 Performance evaluation of stratified random sampling for DQ assessment	37

CHAPTER 4. LABOUR PROTECTION AND SAFETY IN EMERGENCY ..	44
4.1 Introduction.....	44
4.2 Need for guidelines	45
4.2.1. Software quality.....	45
4.2.2 Static analysis	47
4.2.3 Automated static analysis tools	49
4.3 Universal standards.....	51
4.4 Challenges in safety critical systems	52
4.5 Similarities between Different Standards	53
4.6 Conclusion to safety.....	53
CONCLUSIONS	54
BIBLIOGRAPHY	56

INTRODUCTION

The presence of the Internet, social networks, and other technologies, such as the Internet of Things (IoT), has promoted the proliferation of Big Data in all economic sectors. Businesses have the advantage of access to a large volume of data, allowing for more accurate decision-making. This resulted in a caveat, which was the presence of heterogeneous data sources. Data from different sources must be processed and modeled to obtain a standard formulation to better extract information.

This need to unify data from various sources is at the core of collection systems and decision-making. Over the years, the ETL process has been used to address this need for a single source of truth containing well-formatted, standardized, and reliable data on which to perform the analyses. The Transaction Processing Performance Council (TPC) defined ETL as a method of integrating data from various sources in various forms and transforming the data into a uniform model to be placed in a data warehouse. The TPC chose to use the term Data Integration (DI) instead of the abbreviation ETL to give this procedure a more complete nomenclature [3]. Thus, ETL abstracts the heterogeneity of data sources and provides the end user with a single access point to the storage system to submit their queries. According to Sreemathy et al. [6], the entire development process is conducted to meet the objectives of enterprise performance management, application challenge development, finance, and other enterprise management sources. Souissi and Benayed determined that the core element of a business intelligence system that helps managers in their decision-making is the DI [4].

With the growth of big data, ETL has not remained static. This has been described as the ability to have all datasets available at the time of decision making in areas such as the oil and gas industry [1].

De Mauro et al. After reviewing the existing research in terms of defining Big Data, they concluded that the core of the concept is not only expressed by the three Vs of velocity, volume, and variety, but also by technology and analytical methods to clarify some necessary requirements in terms of information usage. They furthered their ideas by adding the concept of transformation into insights and the subsequent creation of economic value as the main way for Big Data to impact businesses [2]. These concepts agree that Big Data represent information assets characterized by the three Vs and

mainly require technological capabilities and resources to enable more advanced transformation to acquire value from the information thus collected. In this research, we argue that the concept of data quality in the processes of this ETL system fulfills the necessary conditions to apply research on Big Data. It is essential to develop algorithms and a correct and robust model because of the amount or volume of data collected and the required technological resources.

The conceptualization of data quality (DQ) has been a subject of research over the years. One of the major studies was conducted by Wang and Strong [18], in which they take the direction of conceptualized data quality as an inherently good product that is tailored to the context of the task clearly represented and made accessible to data consumers.

The necessity for a data integration phase inside the decision system is supported by a number of factors, including diverse data formats, confusion or difficulty in reading data formats, legacy systems utilizing outdated databases, and changing the data source structure over time. What is argued in [19] to make Data Quality assessment difficult is the characteristics of the data sources.

According to Dakrory, Mahmoud, and Ali, the purpose of checking the data quality in the ETL is to guarantee the accuracy of the methods and determine whether or not they need to be changed in order to address the problems. Automating the test procedures to verify the data quality parameters was the goal of their study [20].

In this study, the evaluation of data quality in the ETL service ensures the reliability of the data collected in the Data Management Platform. A variety of data sources results in a diverse range of formats and intricate data structures, making data integration more challenging. Historically, businesses have used only data produced by their internal business systems, such as sales and inventory data. However, the extent of data that businesses currently collect and analyze is beyond this range. The sources of Big Data are highly diverse. The sheer amount of data makes it challenging to determine data quality in a timely manner.

It is challenging to gather, clean, integrate, and obtain the required high-quality data within a reasonable amount of time. Big data is subject to a significant amount of unstructured data; hence, converting unstructured data into structured data and processing the data requires a significant amount of time. [21]

Scientific novelty: Our contribution is to apply a stratified algorithm for data quality assessment, while defining the selection criteria that enable the use of this approach in resource-poor ETL systems.

In a real scenario where not, all data could be stored, the main result of this part of the research was to enable data quality assessment. We also investigate the elements that influence the selection of the reduction model. This study also sheds light on the various problems related to sampling for DQ assessment that still need to be solved.

We also discuss the practical results of using this model. In order to reuse the ideas that emerged in this data management platform framework and make them suitable for any application that uses ETL, we structured the entire system in terms of directed acyclic graphs (DAG).

CHAPTER 1. LITERATURE REVIEW

1.1 ETL

Consider a case study in which a marketing agency launches an advertising campaign for a company or individual associated with the agency. The agency plans to launch the campaign through a network of advertisers. A marketing agency obtains data from various heterogeneous sources to identify the market and conduct a comprehensive study. The objective of this step is to have a unified dataset that allows the campaign to be limited to specific sectors to better target customers. Next, the marketing agency offers a subscription to affiliates that allows them to receive regular updates on customer data (see Figure 1.1). The marketing agency is also in contact with the advertising network to inform them about the cost of the campaign and any other details before asking for feedback.

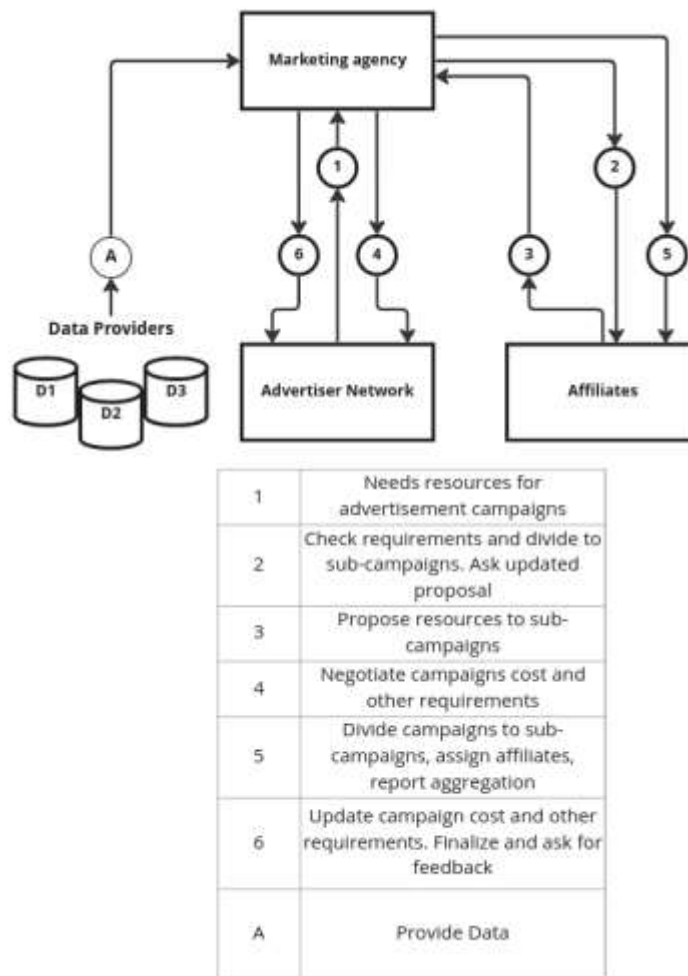


Figure 1.1 - Case study business model

This tedious and complex process demonstrates the importance of the quality of data collected by marketing agencies. The purpose of implementing a marketing data management platform is to make things easier for a marketing agency that uses multiple software platforms to aggregate its advertising network. Second, the implementation of this platform helps solve the problem of the type and format of data offered by data providers to marketing agencies. Third, the implementation of this platform ensures the privacy of the collected customer data. Finally, the customer data is processed, filtered, merged, etc. by the platform before it is forwarded to the marketing agency affiliates.

The above issues meet the conditions for the use of an ETL at the level of collecting the data provided by the Data Providers to the marketing agency.

The data received by the marketing agency's affiliates must go through a stage that allows the evaluation of the quality of this data, as a quality problem will cost huge amounts of money and time to all parties involved.

1.1.1 ETL definitions

Data collection is fraught with difficulty. Some studies, such as [7], have summarized these as follows:

First, because various sources arrange information in entirely distinct schemas, it is critical to transform incoming source data into a common "global" data warehouse schema that will eventually be utilized for querying by end-user applications. Second, operational data suffers from quality issues ranging from simple spelling errors in textual attributes to inconsistencies in values, database constraint violations, and conflicting or missing information; thus, this type of data "noise" must be removed so that end users receive data that is as clean, complete, and truthful as possible. Third, because information in the production systems that populate the warehouse is continually being updated, it is vital to routinely refresh the contents of the data warehouse to present users with up-to-date information. All these difficulties require the data warehouse development team to build the necessary software processes (either manually or through specialized tools) and run them at proper time intervals for the right and full data warehouse populating.

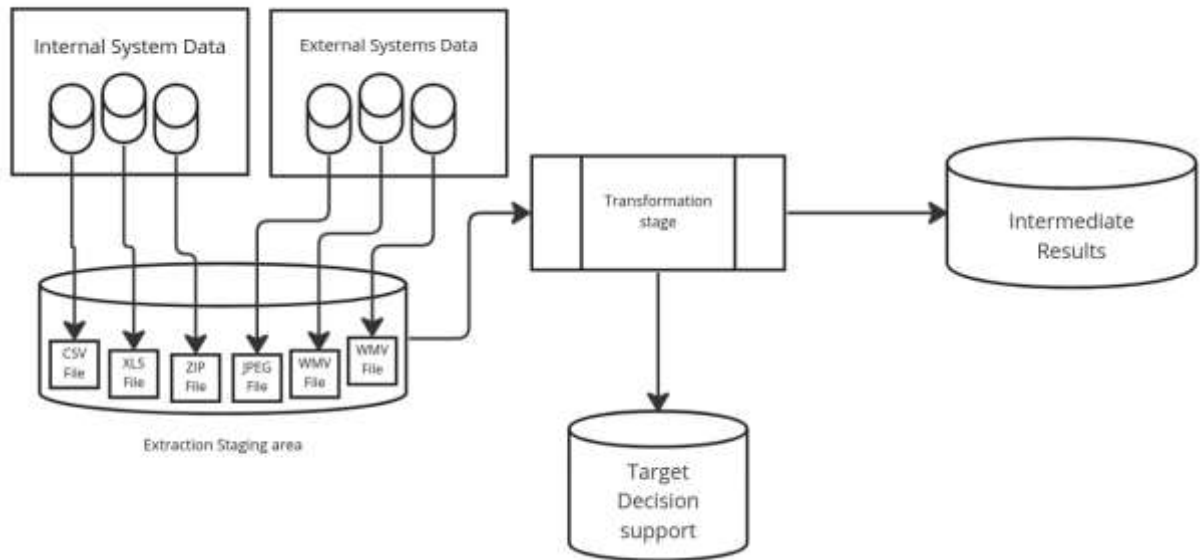


Figure 1.2 - TCP-DI ETL workflow

The primary goal of DI is to collect relevant and transferrable information to assist in highlighting challenges and achieving the advanced vision. Extract, transform, and load (ETL) methodologies are critical in data integration approaches. Companies can utilize ETL to collect data from several sources and combine it in a single, centralized place.[6]

1.1.2 ETL tools review

One of the most popular ETL tools is the Informatica PowerCenter, which is available on premises and allows for connection to a number of database systems, while providing data governance control, master data management, and data masking [6].

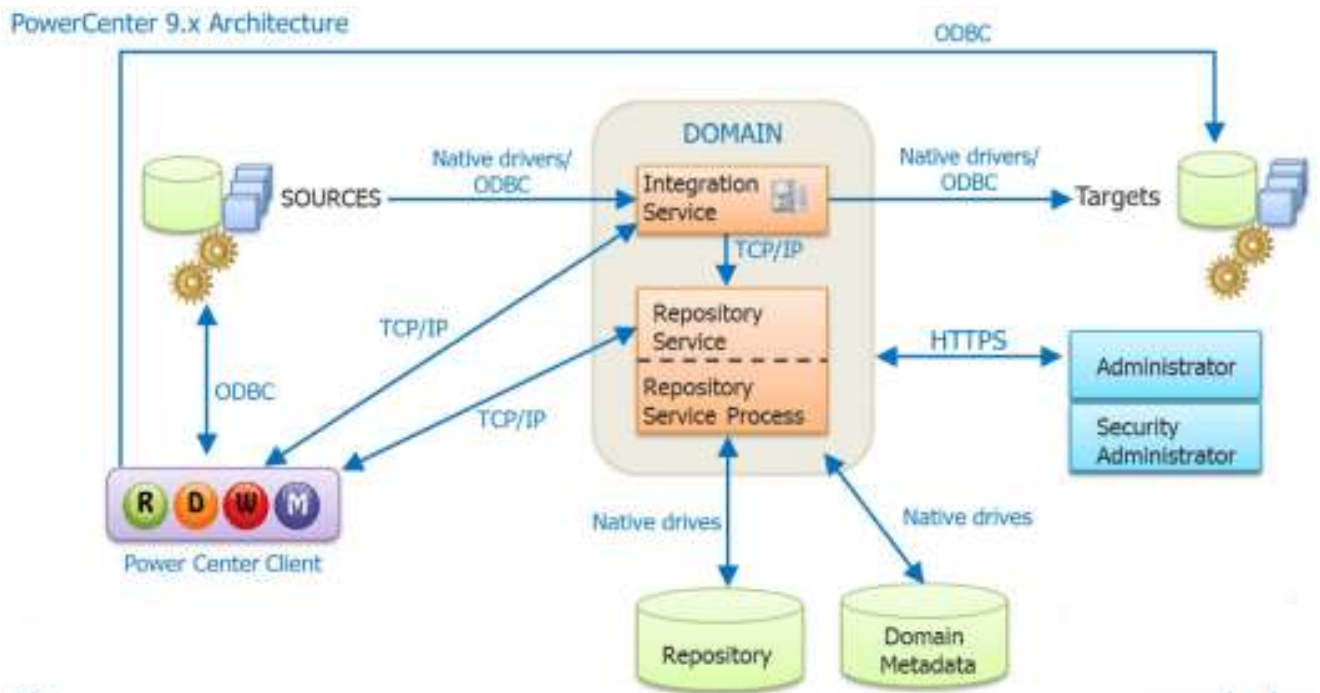


Figure 1.3 - Informatica Architecture

It also allows the visualization of data by connecting them to different sources so that data processing can be performed. Moreover, it provides cloud-based applications and technologies that allow employees to benefit from this network with less effort. Real-time data integration, data analytics, and B2B data fusion are also available, and are some of the benefits offered by this tool. According to research by Sreemathy et al., the Informatica PowerCenter also offers a wide range of features such as data aggregation, semi-structured and unstructured data, and data execution preparation, while also having a metadata feature that helps protect information about the application and data operations.

One tool that has been presented to handle a variety of Big Data is GENUS [4], which has been studied to handle unstructured data (text) as well as image and video data. The principle of data processing in GENUS is to transform the first representation of the data (text, image, or video) into a new representation to load it into a data warehouse. As an output, the tool provides a data warehouse, and in most cases, this data warehouse is then processed by analysis algorithms to extract information requests.

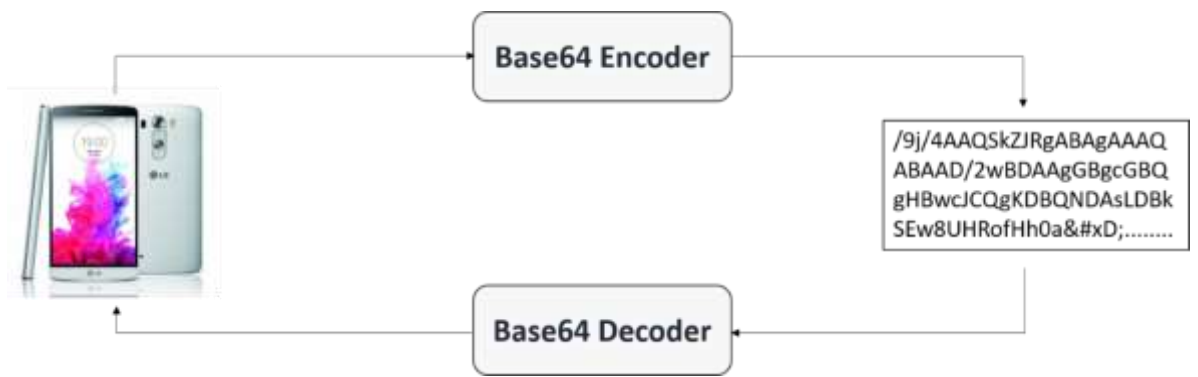


Figure 1.4 - GENUS Image encoding chain

1.2 Data quality

Wang and Strong developed two surveys [18] that were used to collect data from consumers to establish the desired characteristics of data quality. The first survey developed a list of potential traits or characteristics of data quality that the respondents considered when discussing data quality. The second survey assessed the importance of these potential data quality characteristics to consumers. An exploratory factor analysis was conducted using the importance scores from the second survey to produce an intermediate set of data quality dimensions that were meaningful to data consumers.

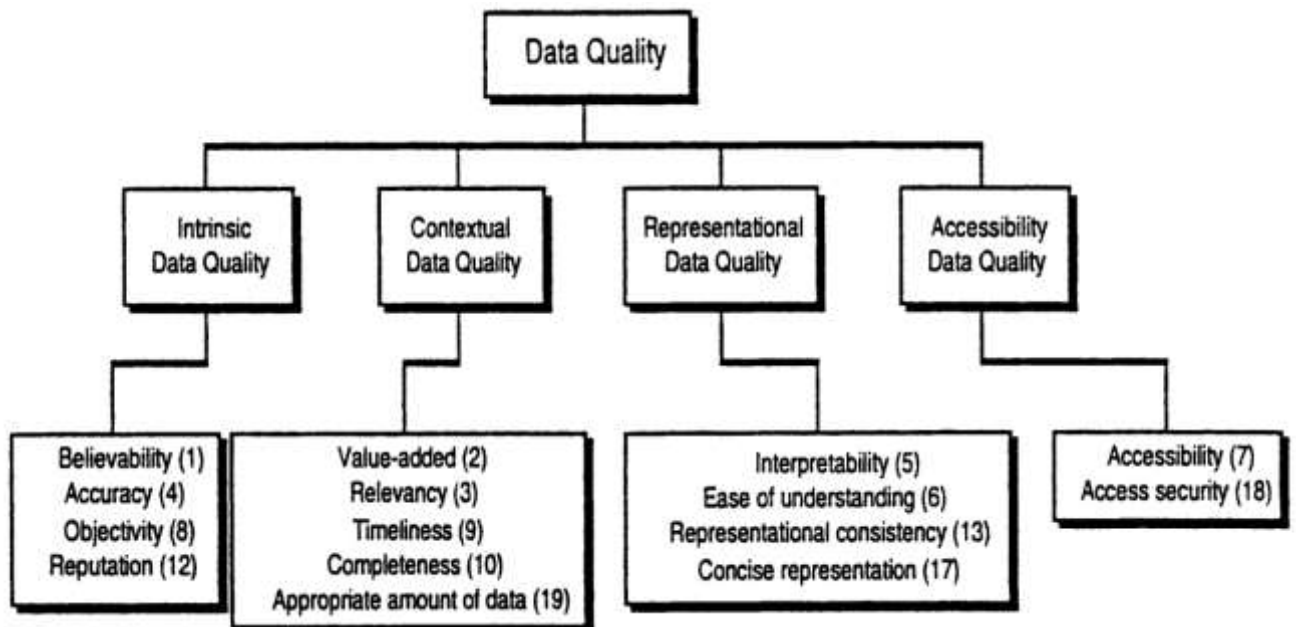


Figure 1.5 - Conceptual Framework of Data Quality

In the first phase, subjects were asked to group these dimensions into categories and then identify these categories. To verify these results, another group of individuals was asked to classify these dimensions into categories in the first phase.

They were able to create a hierarchical structure (Figure 1.5) that encompasses many facets of data quality from the perspective of data consumers, as a result of their research using the two-survey approach.

1.2.1 Data Quality Dimensions

In this study, we considered only the following aspects of data quality: completeness, consistency, uniqueness, validity, timeliness, and accuracy.

Record count validation, data duplicate checking, integrity constraint checking, and data boundary checking are four practical methods for validating data completeness. Validity and accuracy are interrelated.

All values must be constant across all datasets and consistency guarantees this requirement. Field mapping, integrity constraints, aggregation of metrics, and hierarchy-level integrity are common consistency assessments.

Validity, accuracy, and uniqueness are three properties that are consistent with consistency.

Uniqueness guarantees that stored data are free of duplicates. It also provides numerous alternatives for defining its use in an ETL; data duplicates, integrity, and constraints are checked in that order.

The main goal of the dimension known as "validity" is to ensure that the data conform to the syntax (format, type, and domain) of its description. Sara, Tarek, and Abdelmgeid presented three practical methods for assessing the validity: integrity checking, checking the data type of a field, and checking the field length.

To ensure timeliness, it is necessary to ensure that all data are stored for the required period. The aforementioned studies briefly described the following procedures for assessing timeliness: freshness of data and data access. Accuracy: The issue here is to ensure that each piece of information accurately represents the "real-world object or event described. The accuracy of the input data can be assessed using various methods, such as field-to-field comparison, data bounds, and integrity constraints.

1.2.2 Data Quality Objectives in the Context of ETL

One of the most important features of ETL operations is the data quality. Adhianto, Banerjee, Fagan, et al. provided a detailed description of the elements to consider when evaluating the data quality of the ETL process [28]. Data accuracy is the proportion of data free of errors. Data completeness refers to the extent to which the values and entities are not missing. Data freshness is a measure of the timeliness of the data relative to when the target repository for the data source is updated. Data integrity is maintained during transactions and across data sources and the degree to which each user obtains a consistent version of the data. The degree to which consumers can understand the data they obtain is known as the data interpretability.

Cai and Zhu claimed that the key to data quality evaluation is the analysis of each dimension in their effort to create a hierarchical data quality framework from the viewpoint of data consumers. Quantitative and qualitative methodologies comprise the two categories of the present approach. From the standpoint of qualitative analysis to describe and evaluate data resources, the qualitative evaluation technique is based on specific evaluation criteria and needs determined by assessment aims and user requests. It is best for specialists or subject matter experts to perform qualitative analysis. The quantitative method is a formal, impartial, and methodical approach to gathering information that uses numerical data. The elements of this technique, whose evaluation outcomes are more logical and tangible, are objectivity, generalizability, and numbers. After the evaluation, the data were compared with the defined baseline for data quality assessment. A follow-up data analysis phase and data quality report will be produced if the data quality meets the baseline level. Otherwise, new data must be collected if the quality of existing data does not meet the baseline criteria [21].

Other studies [20] used a logical ETL mapping document and metadata repository to evaluate the criteria that should be considered when assessing the data quality. The source database, all applicable intermediate databases, and the DW were first defined as references. This phase includes the metadata repository and the logical ETL mapping document. The path of each file extracted from the source system to its destination is contained in a logic mapping document. The document must be created by a business analyst. This document, also known as a crosswalk or interface design, is an Excel spreadsheet. It is a design that specifies business rules and transformations and evaluates

source or legacy systems. It includes the following fields: Transformation, Source Database, Source Table Name, Source Column Name, Destination Table Name, Destination Column Name, Table Type, and Slow Moving Dimension Type. Finally, a link is created to the metadata store. The required data were extracted from the logic-mapping document and DW metadata and inserted into the database model. The DQ model contained multiple algorithms for each quality metric. Each quality parameter was manually assigned to a set of test routines by searching for those that could identify the quality issues that affected each quality parameter.

1.2.3 ISO Data Quality Standards

According to ISO/IEC 25012, the degree to which the data meets the specifications set by the product-owning organization can be interpreted as the quality of the data product. In particular, these requirements are those that the data quality model reflects through its attributes (accuracy, completeness, consistency, credibility, timeliness, accessibility, etc.).

Table 1.1 - *ISO/IEC 25012 Inherent Data Quality Characteristics*

Characteristic	Definition
Accuracy	The degree to which data has attributes that correctly represent the true value of the intended attribute of a concept or event in a specific context of use.
Completeness	The degree to which subject data associated with an entity has values for all expected attributes and related entity instances in a specific context of use.
Consistency	The degree to which data has attributes that are free from contradiction and are coherent with other data in a specific context of use. It can be either or both among data regarding one entity and across similar data for comparable entities.
Credibility	The degree to which data has attributes that are regarded as true and believable by users in a specific context of use. Credibility includes the concept of authenticity (the truthfulness of origins, attributions, commitments).
Currentness	The degree to which data has attributes that are of the right age in a specific context of use.

DQ characteristics are divided into two main categories by ISO /IEC 25012:

In certain circumstances, inherent data quality refers to the extent to which data quality characteristics have the inherent ability to satisfy both explicit and implicit requirements. **Inherent data quality** refers to the data itself, specifically the data domain values and possible constraints (e.g., business rules that determine the quality required for the characteristic in a particular application), the relationships between data values (e.g., consistency), and the metadata.

System-dependent data quality: this term describes the extent to which data quality is achieved and maintained within a computer system when the data is used according to specified guidelines. According to this view, the technological domain in which the data is used determines the quality of the data. This is achieved through the capabilities of the computer system's components, such as hardware devices (e.g., to make data accessible or to achieve the necessary precision), computer system software (e.g., backup software to achieve recoverability), and other software (e.g., migration tools to achieve portability).

1.3 Tpc-di benchmark

Although it is important to have a highly effective DI system, there has never been an industry standard for evaluating and contrasting its effectiveness. TPC recognized this gap and published TPC-DI, a groundbreaking standard for data integration.

In January 2014, (TPC) released the first iteration of its data integration benchmark, TPC-DI. Their research uses the data integration practices of a fictitious retail brokerage firm to simulate TPC-DI. This involves feeding a decision support system with converted data from various unconnected systems, such as a trading system, internal HR, and customer relationship management (CRM) systems. [3]

Using the cardinality of a certain modeled element in the dataset as the scaling factor, SF helps to understand the amount of data from a particular scaling factor, such as the number of clients or ticker symbols, according to the research in [3] on dataset scaling in TCP-DI.

Therefore, $sizeF(SF)$, where SF is an array-specific factor F, can be used to describe the size F of the input file F at scaling factor SF.

Changes are made to the dimension table when a record with the business key is missing; the inserted record must be given a separate surrogate key value and contain it.

TCP-DI modeling research indicates that it is not possible to start processing a phase until the previous phase is complete to represent the execution rules; the initialization phase is not timed [3]. During the preparation phase, the system must be configured, all the required software components must be installed, and the staging area must be set up. In terms of performance, specifically in terms of scalability, the discussion notes that a workload implementation may have bottlenecks or the system on which it is run may have limitations that restrict its scalability; however, the workload definition should not contain requirements that inherently prevent the scalability of implementations.

With respect to performance, the benchmark Runtime Estimate in this research predicted that it would take between 5 and 10 h to complete a full-volume benchmark.

1.4 Big data

To establish our context, we have briefly defined what the characteristics of data are and why this data is called Big Data. In this section, we will revisit this knowledge by defining in more detail the revisited research that over the years has defined new metrics to consider when defining and characterising Big Data.

1.4.1 Vs and BIG Data

Several studies [8, 9, 10] have been conducted to describe the descriptive characteristics of Big Data. After the introduction of the 3V known as Big Data in the past, Nagham and Laden addressed in their research the concept of seven Vs without forgetting the mention of research done by Khan, Uddin, and Gupta to describe volatility, validity, value, veracity as well as variety, velocity, and volume as the seven characteristics of Big Data. Validity is the use of data for specific and precise purposes. This characteristic is similar to veracity, but is defined separately. The purpose of validity is to ensure absolute confidence in the use of data. As for volatility, this is related to the time of data storage being allowed to provide an overall idea in relation to the archived data and current data. Further research by Rajan, speaks of 10 V are required to characterize Big Data, including volume, veracity, velocity, variety, variability,

volatility, validity, visualization, vulnerability, and value. This definition led to complementary research and introduced this concept to the analytical side of BD.

1.4.2 Batch processing and Big Data

Batch processing refers to modifications carried out on large blocks of data according to Benjelloun et al. Each block is processed independently and separated over time. When the data have previously been saved over time, this type of processing is performed. In essence, batch processing manages beginning and ending jobs [11, 13].

Jobs for batch processing frequently run simultaneously and in succession. Its key benefit is the efficient division of large tasks into smaller tasks. Additionally, it may operate offline, using fewer resources and putting less strain on the CPU. The processor is aware of how long a task will take to complete, what task will come next, and whether execution may be delayed.

MapReduce is the best-known model for this type of processing [12, 13]. The following is a formal definition of MapReduce: The paradigm essentially employs a divide-and-conquer strategy. There are many different commercial use cases that may be addressed using the programming technique known as MapReduce. Breaking the job up into a number of distinct tasks is intended to process massive amounts of data in parallel. The mapper organizes the keys after converting the input data into key-value pairs. Subsequently, based on the key, the reducer combines the data into a single output [13, 14, 15].

The user may automatically distribute large datasets by implementing the two primitives, map, and reduce. Without worrying about task communication or failure recovery, the user can process the data [13, 15]. In addition, it permits data segmentation, resulting in scalability and improved performance.

Chandio, Academy, Tziritas, et al. have been able to determine the importance of the parallel computing paradigm in the context of cloud computing. This paradigm is crucial for solving complex and intractable computer problems.

Bulk Synchronous Parallel (BSP) and Directed Acyclic Graph (DAG) are two current parallel computing paradigms used in cloud computing environments as alternatives to MapReduce. The tasks handled within the framework of these paradigms are computation requests from the end user, and can be divided into many tasks [16].

To develop a basic architecture for Big Data processing, Taleb et al. defined a model consisting of three stages: data generation, data acquisition, and data storage and analysis. In their research, they defined data generation as the phase of creating data from numerous sources, such as sensors used to collect meteorological information or monitoring devices, publications on social media sites, etc., and in their research, they extended data acquisition such as data collection, data transmission, and data preprocessing. With the heterogeneous characteristics of data sources, as well as an unprecedented amount of structured, semi-structured, and unstructured data. The preprocessing of Big Data consists of integration, enhancement, and enrichment, as well as transformation, reduction, discretization, and cleaning of data. Data storage is more related to the infrastructure of the data center where there is storage, and it is spread over several clusters and data centers. A typical example is the use of the Hadoop ecosystem to guarantee the reliability and efficiency of fault-tolerant storage through replication. Data analysis involves the application of algorithms, data mining, and machine learning for the processing and extraction of information required to make decisions [17].

1.5 Sampling for big data

Enormous fault-tolerant storage structures, parallel and graphical processing models, such as MapReduce, Pregel, and Giraph, have been developed as creative solutions to the proliferation of large datasets. However, not every environment can handle this level of resources and not every query requires a perfect answer. This encourages sampling to provide summary datasets that facilitate quick searches and extend the usable life of the stored data. To be effective, sampling must balance the conflicts between resource limitations, data characteristics, and the necessary query accuracy. To enhance the value of the final sample, state-of-the-art sampling goes significantly beyond the uniform selection of items [25].

In big data applications, sampling has become a common method for processing massive volumes of data for real-time analysis. When dealing with enormous datasets, two traditional approaches may be considered: dividing the data into smaller sections for independent study and lowering the number of data columns. An improved UV decomposition method can be used to divide large datasets [26].

In their research [27], Zhang, Zhao, Pang et al. showed that the UV decomposition method cannot reduce large datasets when the dataset is very large.

Sampling is not the only option available; there are other resolution methods in the data reduction category, including dimensionality reduction methods, eigenvalue/vector decompositions, PCA, and SVD, which are often expensive and slow for large datasets. "Sketching" methods for summarizing data streams using hashing and random projections have limited scope and are difficult to capture [25].

1.5.1 A taxonomy for Big Data sampling techniques

This section provides an overview of the sampling methods used. Although we cannot provide an exhaustive list, in this study, we focus on stratified sampling, which is used throughout the research to achieve better performance in performing DQ tasks.

Big-data sampling approaches have been successfully used, as proven by Cormode and Duffiel [25]. They use them in a variety of contexts, such as social networks and network traffic.

The sampling methods developed during this study can be classified into uniform random sampling, two-stage sampling, cluster sampling, systematic sampling, and stratified sampling.

With uniform random sampling, each object of interest has a uniform probability of being included in a sample [32]. The number of objects was determined by the population size. To apply sampling, we must generate a number between one and the size of the population. We then select the objects where the generated number is less than or equal to the number of objects required for the sample plus 1 or 2. A higher number allows us to consider the statistical probability of selecting a smaller number of objects than the one actually needed [31].

Manjunath claimed that two-stage sampling relies on a combined subset of the data group and numerous objects of the same data type as its foundation. This method is employed when information is randomly gathered from dispersed objects or various time periods, and a final sample is then randomly chosen from the combined samples. When several diverse items need to be sampled and the size of the pooled sample is greater than that needed for the assessment, it is appropriate. An appropriate representation of the data for each item or time period was ensured by first-stage

sampling. To prevent any group from skewing the final sample, each sample in the first stage must be proportionate to its subpopulation (of records). The second round of sampling ensured that the combined samples were adequately represented to reflect the complete distribution of data [31].

Cluster sampling is similar to database sampling, in which samples are drawn from fewer subsets, such as specific demographic regions. The subsamples from each cluster were combined to form a final sample. Rather than randomly selecting sales data from each shop, when collecting sales data for a chain of stores, one can select a subsample from a representative subset of stores from each cluster to create a cluster sample. This approach can only be used to represent all retail sales data if the clusters reflect exactly the same relative data types and process consistency. For example, if the files contain the same fields, the processes are the same, and the training and performance measures of the information produces match.

It is recommended to use stratified random sampling when the population to be sampled has a distribution in units such that a small number of units exist for a subtype, according to the study in [31] on the use of random sampling in data testing. In the next chapter, we discuss the mathematical aspects and interpretation of stratified random sampling in more detail.

In addition, the study discusses sampling in which every n th dataset was selected. Using a concrete example, we select a ratio based on the ratio between the total records in the database and the required SSD. The first dataset, which was selected based on the ratio used, was random. If the data population is truly random and organized without bias, systematic sampling is appropriate.

CHAPTER 2. SYSTEM MODELING

In the first two chapters of our study, we describe the context in which we apply data quality assessment, that is, a data management platform with multiple services. The service we focus on is data integration or ETL. Second, we introduce the topic of sampling and its practical use for data analysis and collection. In this section, we illustrate the practical implementation of stratified sampling. The ultimate goal of these results was to integrate them into a platform developed for DQ evaluation.

Because the platform is a commercial product, we developed a reduced prototype model that illustrates the integration of the algorithms developed in this way to allow predictions for the implementation of this system in a commercial product. In particular, we discuss the implementation and performance of Stratified Sampling using test data in an environment with limited processing resources in a reasonable time and without excessive memory consumption. These conditions allowed us to simulate the feeding of Big Data into the ETL system of the developed data management platform.

The goal of this experiment was to ensure the validity of this approach for its implementation in applications that process real-world data. To this end, we evaluated the practical problems that arise in the selection of parameters and their impact on the final result.

To demonstrate the evaluation results, we created several plots that evaluated the performance over time using a linear regression model that allowed us to predict the impact of the implementation in an environment with larger datasets and varying characteristics.

The following subsections provide an overview of the architecture of the data management platform, particularly the ETL part of the system studied in this thesis. The second subsection focuses on the test environment using stratified samples to generate the test data to which the representative DQ assessment is to be applied.

2.1 ETL model

In Figure 2.1, we see the data flow over several edges, with the incoming edge having the task of extracting data from external systems and the outgoing edge having the task of DQ assessment. Most of the data quality assessment functions described in the previous chapters are implemented in the last task before the data are loaded into the data warehouse.

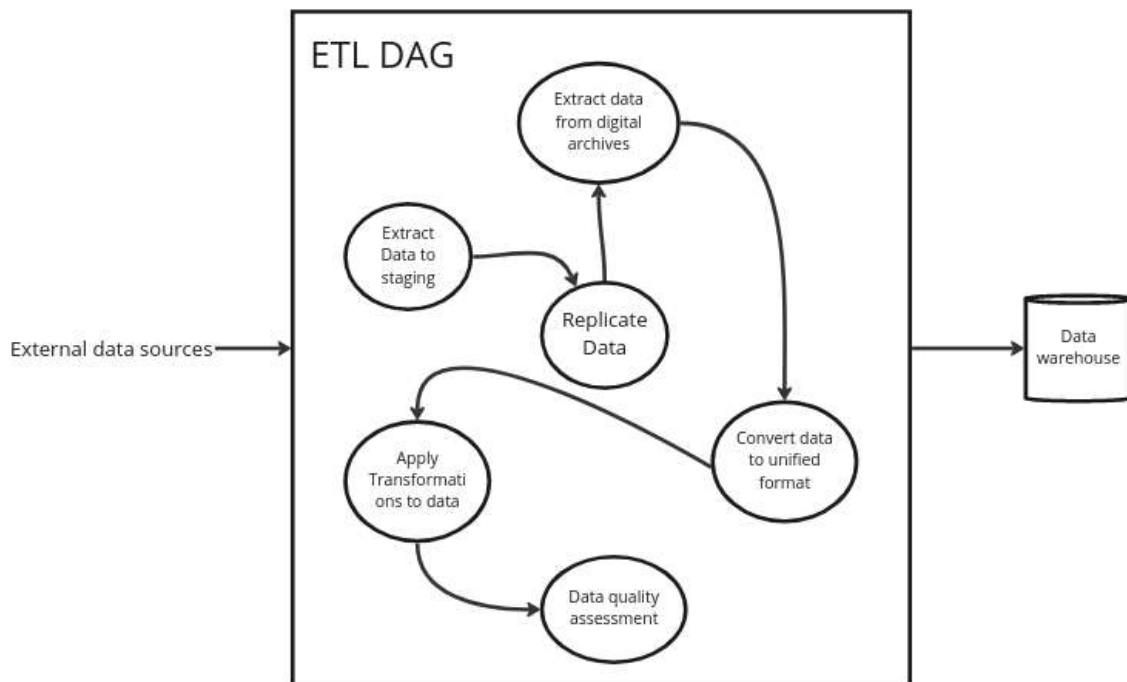


Figure 2.1 - ETL model

The architecture is divided into several tasks, which are implemented in Airflow as DAG to allow more controlled execution. The Airflow tool itself is deployed in a Kubernetes namespace that provides fault tolerance and scalability to the system. The metadata related to all the tasks of the system were collected in a relational database (PostgreSQL). Figure 2.1 shows the overall architecture of the tasks to be executed, making an abstraction of the staging areas between tasks. At the end of the pipeline, the expected result in the data warehouse is a standardized set of parquet files that can be used by the rest of the data management platform.

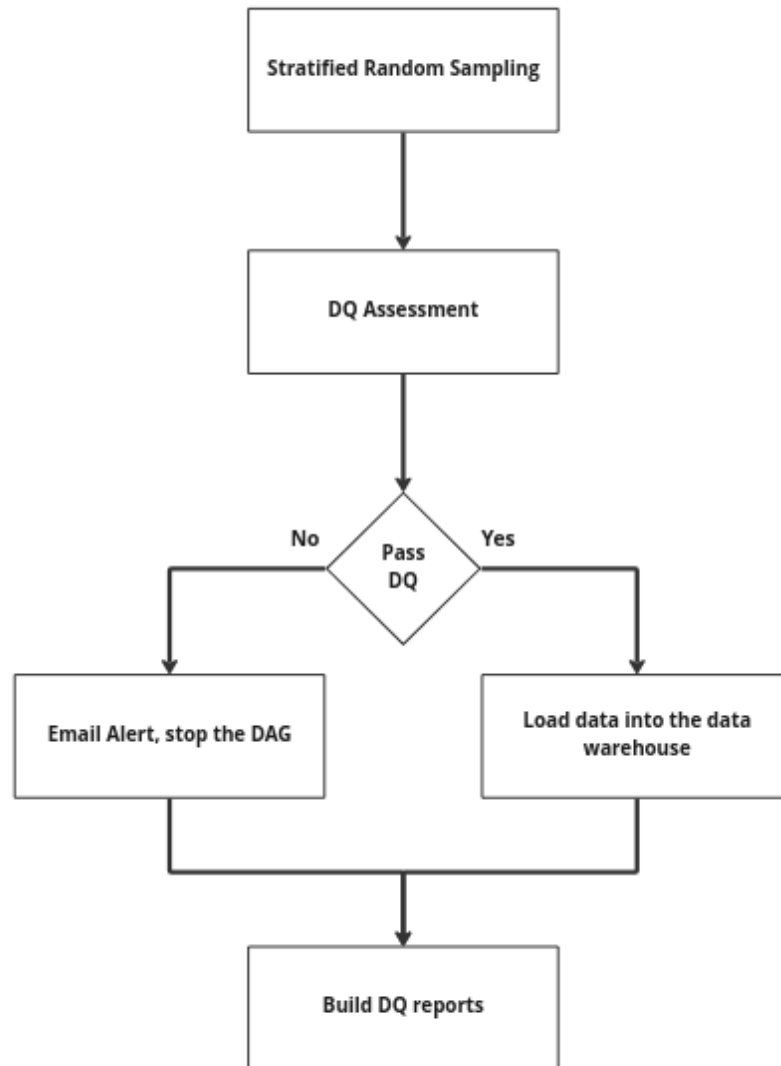


Figure 2.2 - DQ Assessment Task Detailed Sequence Diagram

The first phase is the one on which the main basis of our research is based. It consists of sampling data to allow us to use fewer resources in the next phase. After sampling, the samples were subjected to quality testing (see Figure 2.2). This second step is done with the help of the Great Expectations tool to ensure the quality of the data.

The last phase allowed us to obtain a full report on the quality of our system and the results of the DQ assessment. This is crucial to have a history and to identify the primary causes of the failure or success of data quality tests. Therefore, metadata are collected at the end of the data pipeline in the Metastore.

2.2 System architecture overview

Figure 2.3 represent the overall system architecture

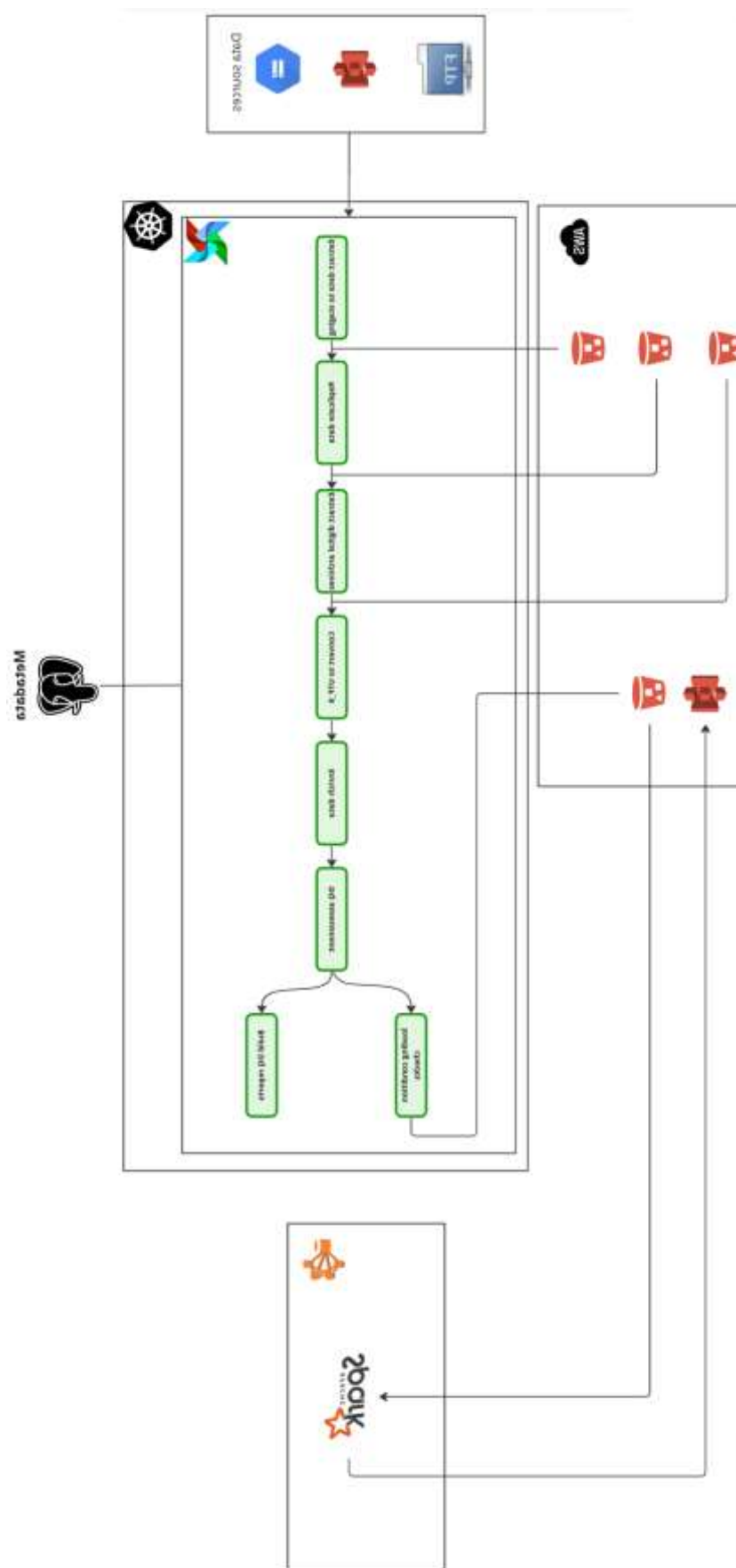


Figure 2.3 - Architecture overview

2.2.1 Metadata store

The Postgres database served as the metadata store in this study. In his quest to create a database system that would work for every application, Michael Stonebraker's most challenging project was Postgres [22].

When Postgres was still in its early design stages in 1990, there were approximately 90,000 lines of C code. The system was used by its "bold and brave" early users [23]. It was created over the course of three years by a team of five part-time students, working under the supervision of a full-time head programmer. Two students in Stonebraker's group, Andrew Yu and Jolly Chen, changed the system's parser to accept an extended variation of SQL rather than the original Postquel language as the Postgres research project came close. Postgres95 was the first Postgres release to support SQL followed by PostgreSQL [22].

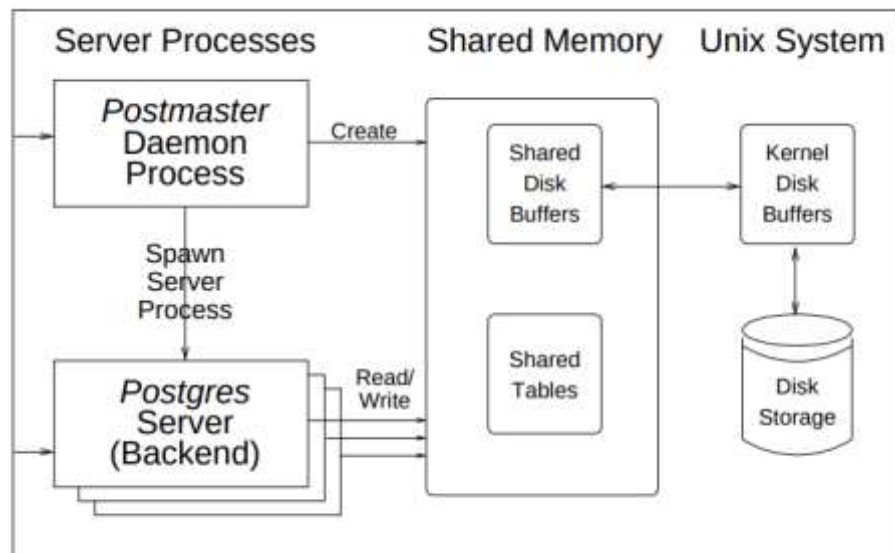


Figure 2.4 - PostgreSQL system overview

According to research [34], database files are accessed through a shared buffer pool. Consequently, the two backends never see inconsistent views of the same file. The Unix kernel frequently includes an additional buffer. Transactions are expected to be atomic, consistent, isolated, and long-lived. Because Postgres does not allow distributed transactions, all statements in a transaction are executed by a single backend. Currently, nested transactions are not processed. Actual insertions/deletions/updates of tuples are annotated as having been performed by transaction N and when completed.

Backends working in parallel ignore changes, knowing that transaction N has not yet been completed. All of these changes become logically visible at the same moment when the transaction is completed. The `pg_log` control file contains two status bits per transaction ID, where statuses in progress, committed, and aborted are possible. Setting these two bits to commit is an atomic operation, indicating that a transaction is committed.

Normally, an aborted transaction changes the status of the `pg` protocol. However, even if the process terminates early, nothing will be lost. When a backend checks the status of this transaction, it finds that it is listed as in process but not running on any backend, which means it has crashed and will update the `pg_log` entry in its name to aborted. No changes to any table file are required during abort [34].

Lane continued to say that Postgres transactions are guaranteed to be atomic only when a disk page is written as an action. This is the case for most current disks when a page is in the physical sector, but most users use disk pages set to 8 K or more, which raises the question of whether writing a page is all or nothing. In any case, the `pg_log` is safe because we only invert bits in the file, and both bits of the transaction state must be in the same sector. However, if we move tuples in a data page, there is a risk of data corruption if a power failure interrupts the page write halfway through (perhaps only a few sectors of the component are written) [34].

2.2.2 Horizontal autoscaling environment

A Kubernetes cluster is composed of a collection of worker computers known as nodes that execute containerized apps. Each cluster has at least one worker node (see Figure 2.5).

The worker node(s) hosts the pods that make up the application workload. The control plane supervises the worker nodes and pods of the cluster. In production situations, the control plane is typically distributed across many computers and a cluster is distributed across numerous nodes to provide fault tolerance and high availability [36].

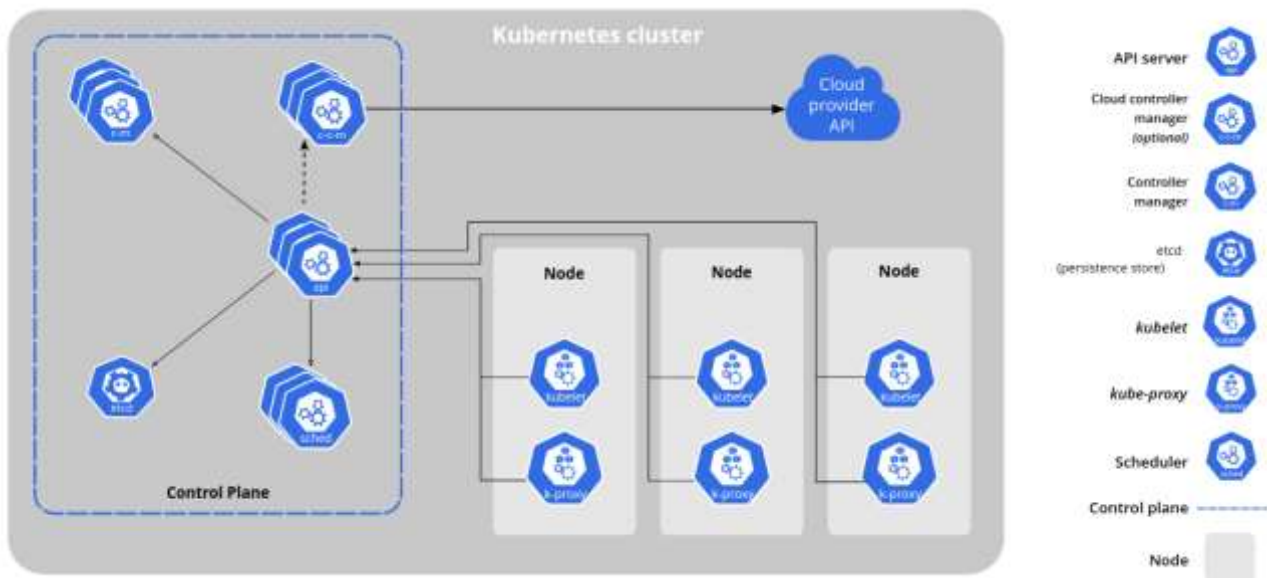


Figure 2.5 - Kubernetes cluster overview

We used Kubernetes to provide scalability and robustness to the entire system in the practical implementation of this research by utilizing the commercial tool architecture. A pod is the most basic unit of execution and resources in Kubernetes, and it comprises a container or set of containers, as well as instructions on how to operate those containers [37]. Each pod represents an application instance and is always associated with namespace. Moreover, the pods from the same application were similar and had the same specifications. A pod can be thought of as a replica in this manner. The intended number of replicas and the amount of resources needed must be provided when deploying an application.

The program, for example, is named Application-A in namespace-1 and requires 250Mi and 250m of accessible memory and CPU for each pod. 'Mi' is an abbreviation for 'Mebibyte,' and 'm' is an abbreviation for 'millicore,' which is a single unit equal to 1/1000 of a CPU core. Kubernetes defines it as a granular technique of measuring CPU resources so that different pods can share a CPU core. Furthermore, within the cluster, each pod is assigned a unique IP address.

Kubernetes can expand apps horizontally because of its architecture. When an application requires extra computing resources, for example, instead of modifying the specifications of current pods, users may simply create another identical pod to share the burden. The IP address of this new pod is subsequently added to the application's service, which distributes incoming traffic to both the new and old pods [37].

2.2.3 Workflow runner

For a repeatable execution of the ETL system with historical control, we have highlighted the Airflow platform that allowed us to build data pipelines in the form of a Directed Acyclic Graph (see Figure 2.6) that contains individual work items called tasks, arranged by considering dependencies and data flows.

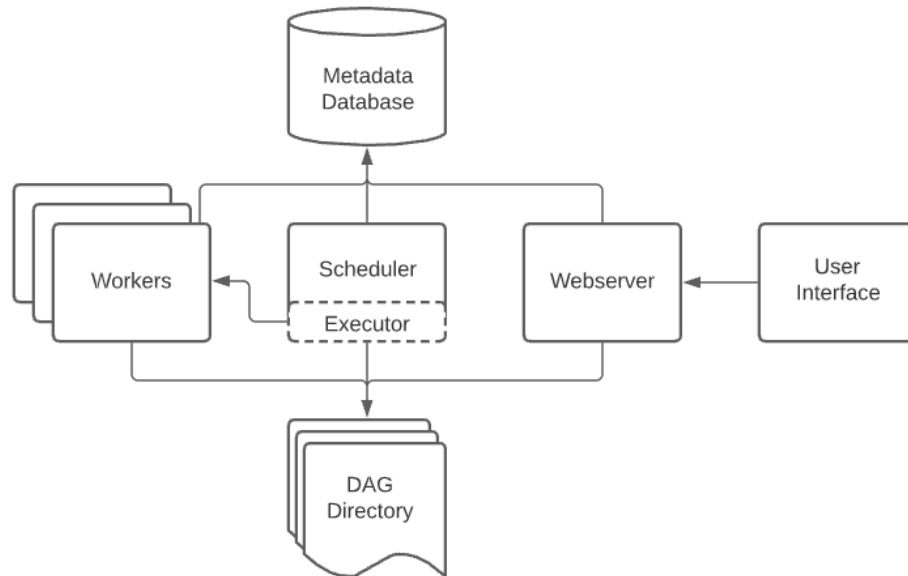


Figure 2.6 - Airflow architecture

A DAG outlines the relationships between tasks, the sequence in which they should be completed, and the execution attempts. The tasks themselves explain what has to be done, such as data extraction, analysis, and activating other systems [35].

Table 2.1 - Types of local test data files

Components	Description
Scheduler	This component is responsible for activating the scheduled processes and simultaneously handing over the tasks to the executor for execution
Executor	The running tasks are managed by the executor, who executes everything in the base airflow installer. Most production-oriented executors push tasks to the workers.

Table 2.1 continuation

Webserver	The web server provides a convenient user interface for analyzing, checking, triggering, and debugging the behaviors of DAGs and tasks.
DAG directory	A folder of DAG files is read by the scheduler and runner (and all workers owned by the runner).
Metadata Database	Stores the state of the scheduler, executor, and web server.

The design of DAGs in Airflow is such that repeated execution is possible and can be performed in parallel. This implies that more data are parameterized by always specifying the intervals at which to execute. We used Kubernetes as the deployment environment in this study (Figure 2.5). Using an Airflow-specific concept called an operator, we can define the tasks and their execution order (see Figure 2.1). The operator used was the `KubernetesPodOperator`, which is customized to the deployment environment.

CHAPTER 3. MEASUREMENT RESULTS

In the previous chapters, we defined the benefits of using stratified sampling and discussed the need for its use in this study to reduce resource consumption in the next task of the DQ assessment (see Figure 2.2). In a practical setting, the DQ assessment task cannot be executed when sampling is not applied. The task timed out or failed completely during the execution in airflow owing to excessive memory consumption. In our study, the data consisted of files stored in a data warehouse. The files are supposed to have a standard model that allows us to perform stratification that is directly proportional to the number of files; thus, the assignment of samples to the different strata is performed in such a way that the overall sample is a faithful representation that allows us to ensure the quality of the overall population data, that is, the set of all records in the files, before the final transfer to the data warehouse.

The population U consisting of N units (total number of combined records of all files stored in the data warehouse before DQ assessment). We define the total number of files K as the number of strata, and the i th stratum U_i consists of N_i units (file records).

$$\sum_{i=1}^K N_i = N \quad (3.1)$$

From the i th stratum a sample s_i of size n_i is selected independently from other strata.

The value y_{ij} is a Boolean DQ assessment of the record y for the j th unit of the i th stratum; $j = 1 \dots N_i$; $i = 1 \dots K$; The i th stratum total Y_i is expressed as follows:

$$Y_i = \sum_{j=1}^{N_i} y_{ij} \quad (3.2)$$

The i th stratum mean \bar{Y}_i :

$$\bar{Y}_i = Y_i/N_i \quad (3.3)$$

The population total Y :

$$Y = \sum_{i=1}^K Y_i \quad (3.4)$$

The population mean \bar{Y} is expressed as follows:

$$\bar{Y} = \sum_{i=1}^K \sum_{j=1}^{N_i} y_{ij} / N = \sum_{i=1}^K W_i \bar{Y}_i \quad \text{With } W_i = N_i / N \quad (3.5)$$

The i th stratum variance $S_{y_i}^2$:

$$S_{y_i}^2 = \sum_{j=1}^{N_i} N_{ij} = (y_{ij} - \bar{Y}_i)^2 / (N_i - 1) \quad (3.6)$$

The population variance

$$S_y^2 = \sum_{i=1}^K \sum_{j=1}^{N_i} (y_{ij} - \bar{Y})^2 / (N - 1) \quad (3.7)$$

3.1 Estimation of the Population Mean

A sample s_i of size n_i is drawn from the stratum U_i with probability $p(s_i)$ according to the sampling scheme $p^{(i)}$. Let $\pi_{j|i} (> 0)$ and $\pi_{jk|i} (> 0)$ be the inclusion probabilities for the j th unit and the j th and k th ($j \neq k$) Units in the i th stratum, respectively. Based on the selected sample s_i , an unbiased estimate for the mean \bar{Y}_i of the i th stratum is given by:

$$\hat{\bar{Y}}_i = \sum_{j \in s_i} b_j(s_i) y_{ij} \quad (3.8)$$

Where $b_j(s_i)$'s are constants free from y_{ij} 's and satisfy the unbiasedness condition:

$$\sum_{s_i \ni j} b_j(s_i) p(s_i) = 1 / N_i \quad \text{for } \forall j \in U_i; \quad i = 1, \dots, N. \quad (3.9)$$

3.2 Performance evaluation of stratified random sampling for DQ assessment

The goal of the experiment was to prove the effectiveness of data sampling for quality assurance. To this end, we modeled a test environment that differed from a commercial product to allow flexibility in performing the sampling task as well as various performance experiments. The configurations of the computer used to test the Stratified Sampling algorithm had 8 GB of RAM. The studied data occupy approximately 93.8 GB of disk space, and the CPU, a product of the manufacturer Intel Corporation, has two cores with a size of 1487 MHz and a capacity of 4100 MHz

Table 3.1 - *Types of local test data files*

Types of files	Size (bytes)
text/csv	14,243,042
text/csv	97,276,566
text/csv	4,020,757
text/csv	61,913,651
text/csv	7,883,148
text/csv	89,328,022
text/csv	1,380,690

The results of the tests without sampling resulted in a timeout as the execution time exceeded the normal values. Therefore, the task could not be completed, and the memory and CPU resources were insufficient to complete the execution.

Therefore, we divided the DQ task into two tasks. First, we introduced sampling, as described in the previous sections, and then evaluated data quality (see Figure 7). In this section, we present the basic results of the Stratified Sampling execution time in the test environment. This provides the necessary conditions and information for adapting

the execution to the production conditions of data integration systems working with Big Data in batch processing.

	stratumnum_of_units
0 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file1.csv	3
1 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file2.csv	20
2 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file3.csv	1
3 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file4.csv	14
4 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file5.csv	2
5 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file6.csv	21
6 /home/aser/Desktop/BIG_FILE/files-20221007T134418Z-001/files/file7.csv	1

Figure 3.1 - Sample size allocation

We define the weight that determines the number of samples to be collected from each stratum as a value directly proportional to the size of the data and inversely proportional to the estimate of the weight of the expected global sample. As shown in Fig. 3.1, the distribution of units in the strata is directly proportional to the weight of the test data.

	time_seconds	mem	weights	sample_size
0	0.2870	5263	3.622586e-07	6
1	0.4845	6957	1.811293e-06	8
2	0.7064	9488	3.622586e-06	11
3	0.9429	12091	5.433879e-06	14
4	1.5004	16360	7.245173e-06	19
5	1.5622	18909	9.056466e-06	22
6	1.8209	21501	1.086776e-05	25
7	2.1004	25748	1.267905e-05	30
8	2.3788	27614	1.449035e-05	32
9	2.9125	32685	1.630164e-05	38
10	2.7981	34348	1.811293e-05	40
11	3.1339	38733	1.992422e-05	45
12	4.1452	42024	2.173552e-05	49
13	3.9346	45369	2.354681e-05	53
14	4.1216	47263	2.535810e-05	55
15	4.6825	50518	2.716940e-05	59
16	4.5302	52451	2.898069e-05	61
17	6.0136	61811	3.260328e-05	72
18	5.7821	65101	3.441457e-05	76
19	5.8708	68507	3.622586e-05	80

Figure 3.2a - Local test results part 1

20	6.7806	72006	3.803716e-05	84
21	6.6961	75207	3.984845e-05	88
22	6.6403	77091	4.165974e-05	90
23	7.2159	82028	4.347104e-05	96
24	7.2434	83148	4.528233e-05	97
25	7.7076	87349	4.709362e-05	102
26	8.0391	89260	4.890491e-05	104
27	8.9160	93343	5.071621e-05	109
28	8.7984	96013	5.252750e-05	112
29	9.1002	99298	5.433879e-05	116
30	9.1723	103590	5.615009e-05	121
31	9.2008	106093	5.796138e-05	124
32	9.2773	108933	5.977267e-05	127
33	10.0598	111307	6.158397e-05	130
34	10.1254	113958	6.339526e-05	133
35	10.3715	120569	6.520655e-05	141
36	10.4451	123247	6.701785e-05	144
37	10.7970	125790	6.882914e-05	147
38	11.9324	129251	7.064043e-05	151
39	12.0417	131929	7.245173e-05	154

Figure 3.2b - Local test results part 2

We can observe a direct correlation between memory and the choice of weight used in determining the sample to be included. Figure 3.2a and Figure 3.2b show the results of sampling with the weight setting versus the choice of units to be contained in the strata. We can also conclude that, in spite of a linear correlation between all the dimensions of performance, there is an innumerable advantage for the choice of the weight in the example quoted above. We can observe a reduction of 99,9 % in the size of the data for the constitution of the sample, and the time of execution of the sampling is included between 0.2 and 12.04 seconds. Contrary to the use of all the data, sampling proves to be efficient; nevertheless, this strategy presents a disadvantage that is linked

to the choice of the weight, and it may not allow a not too representative sampling of the data in the strata. However, as the weight increases, the execution time also increases.

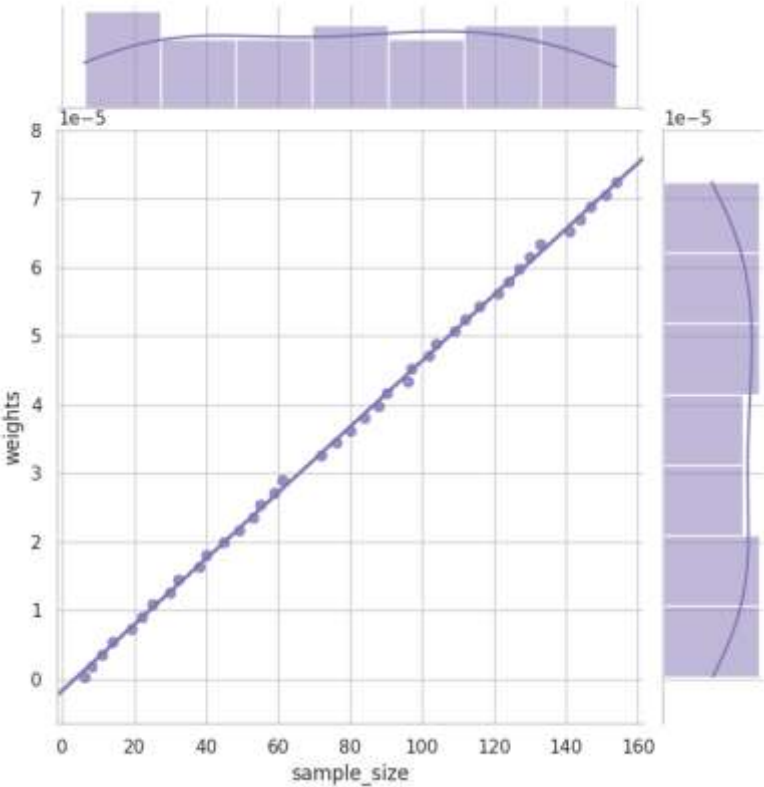


Figure 3.3 - Sample data size by varying strata weights from $3.62 \cdot 10^{-7}$ to $7.62 \cdot 10^{-5}$.

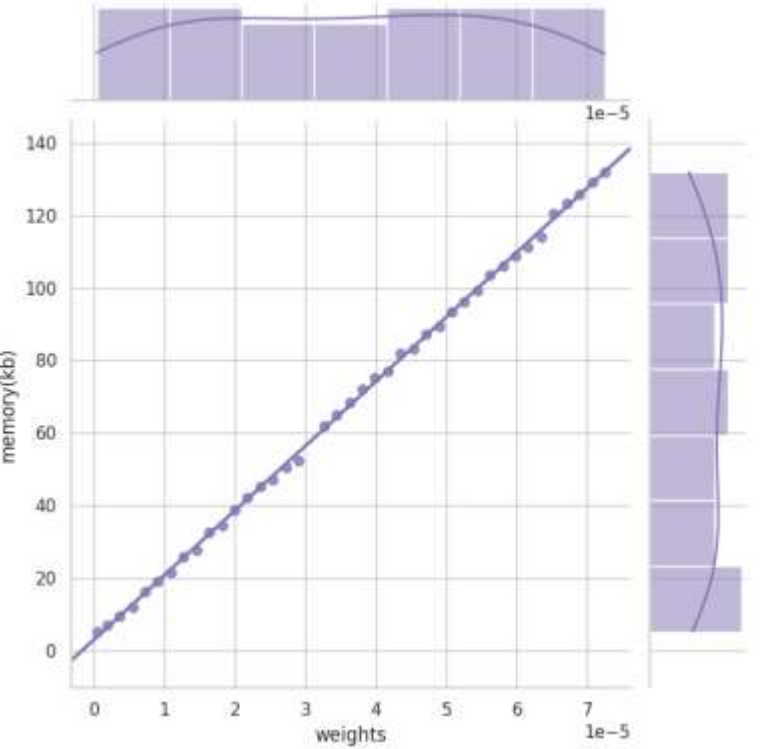


Figure 3.4 - Memory consumption by varying strata weights from $3.62 \cdot 10^{-7}$ to $7.62 \cdot 10^{-5}$.

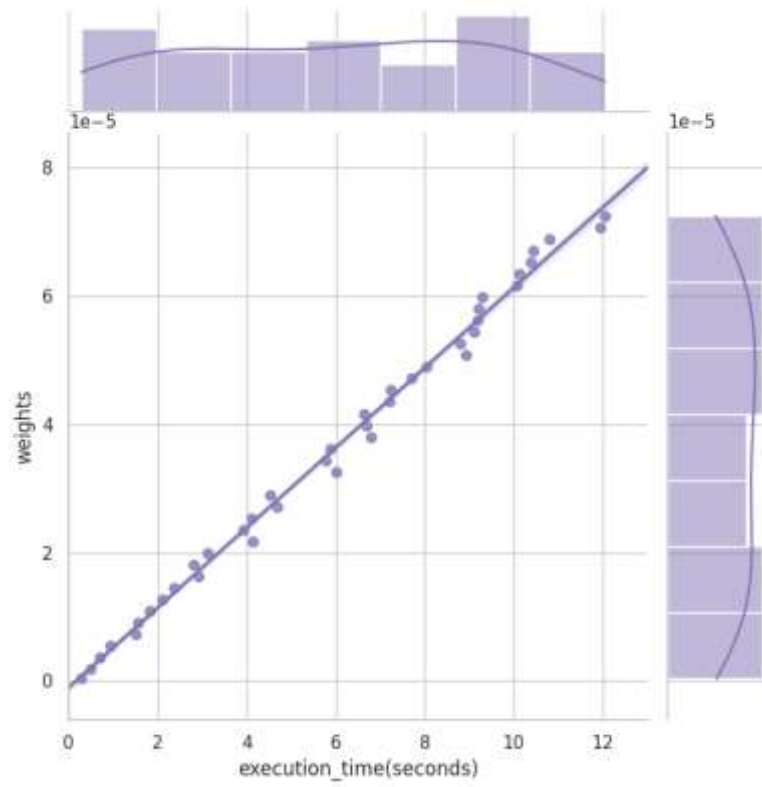


Figure 3.5 - Weight choice versus execution time.

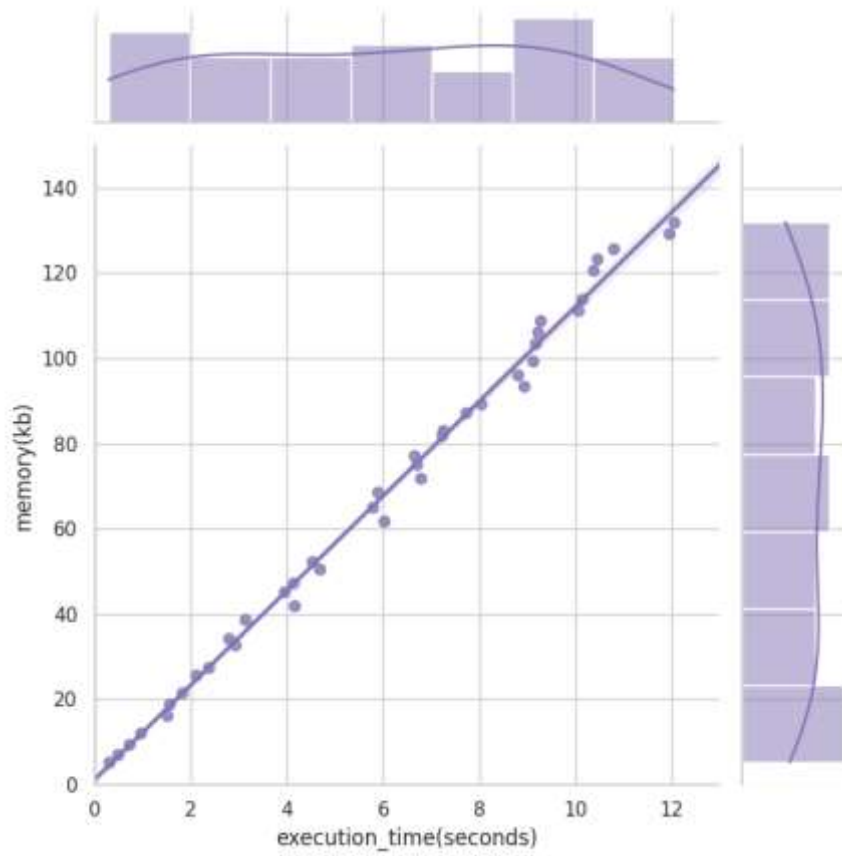


Figure 3.6 - Weight choice versus execution time.

We first varied the size of the data from 1,380,690 B to 97.27 Mb and then compared the seven cases where choosing the same weight resulted in a random number of records to process relative to the size of the file. As long as the value of the weight was above a threshold higher than $3.62 \cdot 10^7$, there was variability in the distribution of the total records for each layer; however, in the opposite case, each layer produced a single record. As shown in Figure 3.2, finding the most diverse records in the final sample is directly proportional to the chosen weighting. Therefore, in an environment involving data with high variability, it is more advantageous to choose an equally high weight to accurately represent the data. The choice of a high weight has several consequences, as shown in Figure 3.3, which shows a linear correlation with the variation in the weight point. This is a serious problem for the relationship between volume and variability, leading to the need to balance the choice of weight. We determined the execution time by calculating the time elapsed between the application of simple random sampling to obtain the number of records to be included in each stratum and the combination of all the strata. We also found a correlation between the weighting and execution time, as shown in Figure 3.4, and concluded that the correlation between the two entities is beneficial because we can infer that reducing the weighting directly affects execution time. In a resource-constrained environment, where we aim for normal execution time without compromising the quality of the samples, it is necessary to choose a minimum weight without compromising the variability of the data. For memory consumption varying between 5263 B and 131,929 kb, we observed a linear increase in execution time from 0.28 seconds to 12 s throughout the process with the test setup environment (see Figure 3.5). This allowed us to determine the reliability of our process compared with an environment with a higher data volume. The problem that arises is the reduction in execution time without reaching an unattainable memory consumption. Therefore, before applying Stratified Sampling in a practical environment, it is of utmost importance to test and determine the applicable thresholds for weight, memory, and execution time on a small scale before determining applicability in a production environment.

CHAPTER 4. LABOUR PROTECTION AND SAFETY IN EMERGENCY

Based on Safety critical software ground rules by Addagarrala and Kinnicutt

Safety critical software development field is one of the active research areas in many industries like automotive, medical, railways, nuclear and aerospace are placing increased value on safety and reliability. Safety critical software systems are those systems whose failure could result in the death or a serious injury to the people's life, security is one of the important topics in the field of safety-critical systems and it must be addressed completely in order to operate safety critical software successfully. In this paper we present a study about the set of standards and different ground rules to be followed in critical software development practices in different industries and the challenges in applying these standards. We also discuss the role of static analysis and software integrity levels in these standards, similarities in these standards and the set of activities followed in the development process of these standards.

4.1 Introduction

Safety critical systems in the automotive industry are life critical systems which if malfunctioning may result in death or serious to human life. Due to these significant costs, safety critical systems must be designed, implemented and tested to ensure robust, efficient performance and no potentially hazardous software bugs. The C programming language is used commonly in automotive safety systems because it executes quickly, but it is a language prone to errors. Many of the problems with using C in embedded systems arises from memory management errors from pointer misuse or buffer overflows. Another main difficulty with the C programming language lies in the differences in compiler implementations of the language grammar, leading to executables with different behaviors based on the compiler options used and the different architectures of embedded systems. Standardized processes have been developed by standards bodies such as ANSI. Many of the vulnerabilities possible in C can be mitigated through the use of well-designed programming rules. Developing software for safety critical systems needs to consider all aspects of security and quality.

The safety integrity concept grew from development of safety critical systems in various industries such as the automotive, aerospace, medical and railway industries.

The safety integrity concept was first introduced by the IEC 61508 standard and later it was taken up and inherited in various offshoot standards. “Safety” as used in the safety critical software refers to developing software to prevent harm or catastrophe from happening. The safety integrity concept comprises two components:

- integrity against random failures;
- integrity against systematic failures.

The main difference between the two are the systematic failure cannot be quantified by the way of probabilistic computation and they mainly occur due to human errors during the different phases of software development process. Random failures result from hardware malfunctions and they occur randomly over time, due to aging and wear and tear on the hardware. Because of the nature of software, software applications are not subjected to random failure. In this paper we present an overview and analysis of a set of good standards developed in different industries for the development of different safety critical software systems, as well as a list of static analysis tools used and their role in developing high-quality code.

4.2 Need for guidelines

4.2.1. Software quality

In general, people increasingly rely on more safety critical software systems in their mode of transportation, so developing such safety critical software always to be correct and perceived to be correct becomes more important to help ensure that catastrophes do not happen. In order to ensure the embedded software is correct, a unified approach is needed in software development with agreed standard techniques across any industry. In the automotive industry as an example, one safety critical system installed in the vehicle may include braking and controlling a particular function like antilock braking during emergency stops. These braking components (hardware and software) are supplied by the original equipment manufacturer or a third-party entity. Normally in automotive industry, most software developed as a part of the entire system is embedded software. Every vehicle manufacturer will specify some system specifications which ensure the following requirements:

- a set of interfaces to communicate with the sensors and other vehicle components;
- the functional performance of the software;
- external environmental requirements such as climatic extremes and electromagnetic compatibility.

Software is considered one of the major components in auto- motive industry. When we compare the software with other hard- ware components we can find some similarities and differences between them:

Similarities:

- both may be subject to continual improvement and development;
- both should be subject to strict quality control procedures; and.
- specialist skills are required in its development.

Differences:

- errors in the software are systematic, not random;
- software is considered intangible;
- software is perceived to be easy to change.

Any embedded safety critical software developed should undergo proper software development practices. Procedures and standards must be followed during the development and vali- dation of software in embedded systems to efficiently improve and maintain quality control. MISRA (Motor Industry Software Reliability Association) developed the first standards in November 1994; up to that time, no specific standards or guidelines existed in national or international vehicle software development. The primary reason behind the development of these standards is that every critical system development has standards and their integrity levels more strict than other software systems, and the automotive test environment uses many vehicle components and simulations to test systems and software extensively before they reach the customer. This led to the development of standards for vehicle-based software systems.

Coding standards are used to improve software reliability and security. These coding standards include the set of rules that help the developers avoid dangerous language constructs; they also help limit the complexity of functions and maintain the standard coding structure by following the consistent syntactical and commenting styles

specified in the standard. These coding standard rules help to reduce the occurrence of flaws and make it easier to maintain and test the software, as the code becomes more readable and is better documented.

It is very common that as the coding standards improve over time, it includes a set of rules whose objective is to accomplish human code reviews. During code review, developers try to improve the software quality prior to deployment by examining the code, fixing potential bugs and ensuring the coding standards are met. Developers use a set of static analysis tools during the code review that helps them determine whether any set of warnings may be related to code style or design or documentation. The role of static analysis tools is to analyze the source code, with the aim of complementing the compiler by highlighting potential issues that may arise in the software system like uninitialized variables, poorly commented code, etc.

Compilers and other link chains like linkers/loaders by default often emit warnings rather than halt a build with a fatal error in the event of uninitialized variables or other potential issue. A warning during compilation is an indicator to the developer that a construct may be technically legal but questionable, or may be exercising a corner of the language that is not well defined. Such constructs are frequently the cause of subtle bugs. To ensure that developers do not intentionally or accidentally ignore warnings, the compiler can be configured to treat all warnings as errors. Many compilers have such an option.

4.2.2 Static analysis

The software source code can be analyzed with static analysis using manual or automated methods. In manual analysis either a checklist or coding standards are used, while static analysis tools are used in the automated approach. The main objective of analyzing software is to ensure the absence of bugs in the software [11, 15]. Some coding standards used in the manual analysis are: MISRAC/C++, GNU Standards-C, JSFC++, CERT-java, JPL, Neutrino, RUNTIME, CERT, CMSE, CON-FORM, CWE, DERA, and EADS. Motor Industry Software Reliability Association (MISRA) is an organization that provides guidelines for embedded software development to support electronic components used in the automotive industry. The main intention of MISRA is to give assistance to the automotive industry to make vehicle systems reliable and secure. These

guidelines were geared towards the use of the C programming language in vehicle-based software systems. The MISRA Guidelines are intended to achieve the following objectives to assure safety, robustness and security to the software, and minimization of both accidental and regular faults in the system design.

Currently, the MISRA standards are meant for the C and C++ programming languages. The first MISRA standard was issued as MISRA C:1998; this first release disseminated a set of guidelines for the use of the C programming language in vehicle-based software. MISRA C:1998 has 127 rules, of which 93 are required and 34 are advisory; these rules are numbered in sequence 1 to 127 [15]. The second edition of the standards released was MISRA C:2004. These guidelines are used in safety critical systems; it contains 142 rules, of which 122 are required and 20 are advisory. Most of these guidelines can be reviewed using static analysis tools, while the remaining rules may be reviewed using dynamic analysis tools. For good software design of safety critical systems, both required and advisory rules must be considered in all projects even if they are not fully MISRA-compliant. The required rules must be implemented by developer, and the advisory rules should also be addressed or examined even though it's not compulsory in the standards. Several selected rules are normally not checked by the compiler, as shown in Table 1.

We conducted several experiments using these rules with some test results. These experiments were conducted using the lint static analysis tool and IDEAS development environment as well as the MISRA C:2004 standard for a Chrysler project. To explain what types of violations are specified and how the violations can be corrected, we have taken two required rules from MISRA to concentrate on.

Rule: 10.1

The value of an expression of integer type shall not be implicitly converted to a different underlying type if:

- 1) It is not a conversion to a wider integer type of the same signedness;
- 2) The expression is complex;
- 3) The expression is not constant and is a function argument; or
- 4) The expression is not constant and is a return expression.

Here is an example:

```
UInt8 a = 0xffU;
```


uint8 b = 0u;

uint16 c = 10u;

b = b + 5; /* not OK, 5 is signed */ b = b + 5U; /* OK, same signedness */

The test result using Qtool is shown in Fig. 4. The MISRA 10.1 rule violation report is shown in Fig. 4.1, while the violation code and the resolved code are shown in Fig. 4.2 and 4.3, respectively.

Rule12.5

The operands of a logical && or || shall be primary-expressions.

For example,


```
if ((x>c1) && (y>c2) || (z>c3)) /* not OK */ if ((x>c1) &&
((y>c2) || (z>c3))) /* ok extra braces () used.
```

Note the extra parentheses () used to explicitly specify the order of precedence for the logical or operation. The result of a MISRA 12.5 test violation is given in Fig. 2.

Again, this code snippet shows that the resolved code contains explicit parentheses in the logical expression to make it clear what the order of operations is intended to be.

4.2.3 Automated static analysis tools

MISRA is an organization with a lot of influence in the software development of automotive software systems. Members of MISRA include but are not limited to: Bentley Motors; Delphi Diesel Systems; Ford Motor Company; and Jaguar Cars Ltd. Because of their influence, several software vendors sell analysis tools that support the MISRA standards;



1550	app_LowPowerLayer.c	Violates MISRA 2004 Required Rule 10.1, Implicit conversion changes signedness	499	15	for(lub_j=0; lub_j < MAX_DIGITAL_WAKEUP_INP; UTS; lub_j++)	PCLint	960	M10.1	Req
------	---------------------	--	-----	----	--	--------	-----	-------	-----

Figure 4.1 - (A) Qtool report for the MISRA 10.1 violation

```

491 static uint8 LP_poll_digital_input( void )
492 {
493     IoHwAb_BoolType digital_value;
494     uint8 have_to_wakeup;
495     uint8 lub_i;
496
497     have_to_wakeup = 0U;
498
499     for(lub_i=0; lub_i < MAX_DIGITAL_WAKEUP_INPUTS; lub_i++)
500     {
501         raub_have_to_wakeup_digital[lub_i] = FALSE;
502     }

```

Figure 4.2 - (B) MISRA 10.1 violation code snippet (before change)

```

491 static uint8 LP_poll_digital_input( void )
492 {
493     IoHwAb_BoolType digital_value;
494     uint8 have_to_wakeup;
495     uint8 lub_i;
496
497     have_to_wakeup = 0U;
498
499     for(lub_i=0u; lub_i < MAX_DIGITAL_WAKEUP_INPUTS; lub_i++)
500     {
501         raub_have_to_wakeup_digital[lub_i] = FALSE;
502     }
503

```

Figure 4.3 - (C) MISRA 10.1 violation code snippet (after change)

Some of these vendors include Gimpel Software, Axivion, Cosmic Software and Green Hill Software, among others. Two popular automated static analysis tools are PC-Lint and RSM (Resource Standard Metrics). This section summarizes these tools.

PC-lint

The PC-lint is a static analysis tool. It will check the source code of C/C++ and figure out the bugs, inconsistencies, non-portable constructs, redundant code, etc. It is developed by Gimpel Software and it has been continuously maintained for more than 25 years.

Two examples demonstrating the types of violations PC-Lint can catch are:

a) The goto keyword shall not be used, The PC-Lint can be configured to generate a warning message each time the goto keyword appears in your C/C++ code by including deprecate (keyword, goto, violates coding standard) in our local lint configuration file.

b) Comments Shall never be nested, The PC-Lint generates an error whenever such comments are found in the source code.

4.3 Universal standards

Different sets of standards have been developed for safety critical software systems in several industries such as railroad, medical, and aerospace. Table 2 shows the industry and the corresponding standard that applies to the industry.

Table 2: Industries and Applicable Safety Critical Software Standards

Industry	Standard
Automotive	ISO26262
Aerospace	DO178B
Medical	IEC62304
Railway	EN50128

ISO26262 automotive functional safety standard

The ISO 26262 discusses the importance of an automotive specific international standard that focuses on the safety critical components. It is a derivative of the IEC 61508, the generic functional safety standard for electrical and electronic systems. The high increase of complexity in the automotive industry resulted in the industry putting significant efforts to provide robust and responsive safety compliant systems. ISO 26262 uses a set of steps to manage the functional safety and to regulate the product development on a system at both the hardware and software levels. The main goal of the ISO 26262 is as listed below.

- a) It provides an automotive safety lifecycle (management, development, production, operation, service, decommissioning) and supports tailoring the necessary activities during these lifecycle phases;
- b) It provides an automotive specific risk-based approach for determining risk classes (Automotive Safety Integrity Levels, ASILs);
- c) Covers functional safety aspects of the entire development process (including such activities as requirements specification, design, implementation, integration, verification, validation, and configuration); and

d) It provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety is being achieved.

Here in this paper, we mainly discuss the product development at the software level and safety integrity levels as defined by ISO 26262. The ISO 26262 specifies the set of steps needed to be considered for the product development at the software level in the automotive industry to provide the safety required. Steps discussed in ISO 26262 include.

- a) Requirements for initiation of product development at the software level;
- b) Specification of the software safety requirements;
- c) Software architectural design;
- d) Software unit design and implementation;
- e) Software unit testing;
- f) Software integration and testing; and
- g) Verification of software safety requirements.

4.4 Challenges in safety critical systems

In one or the other way for people in the software community working on Safety Critical Systems technology, safety critical system is an application where human safety depends on the correct operation of the application. , as we also briefly describe below. The important point in safety critical systems is security and it must be named in order to work successfully. The major difficulty here exists very much in the software engineering than security. Many security difficulties that arise in network information systems appears because of software defects make the systems weak to attacks. Even now such attacks exist's because system continues to be deployed with vulnerabilities. In some cases, what amounts to completely new technologies are required. The number of interacting safety-critical systems present in a single application will force the sharing of resources between systems. This will eliminate a major architectural element that gives confidence in correct operation-physical separation. Knowing that the failure of one system cannot affect another greatly facilitates current analysis techniques.

4.5 Similarities between Different Standards

EN50128, DO-178B, and ISO26262 are derived from the IEC 61508 standard (General Electrical/Programmable electronic devices). The considerable similarities between these standards. All these standards are offering the guidance for core software development process. The EN50128 process is based on waterfall and V model whereas the ISO 26262 describes the software lifecycle under V-model framework and it is allowed to use the agile development for all the standards. Related to the safety integrity levels all these standards use the different terminology like Safety integrity levels (SIL), Automotive Safety integrity Levels (ASIL) to check the safety levels. With the higher safety systems need more checks and high control.

4.6 Conclusion to safety

This paper is meant to provide sets of standards and guidelines followed in different industries in the field of safety critical software development. We hope the paper can help as a point of reference in the automotive, medical, aerospace, and railway industries for the developer creating embedded software to get an overview about the development process and SIL involved.

CONCLUSIONS

Data quality assessment and assurance have been identified as important issues in decision-making systems, particularly ETL systems. This study provides an overview of the definition of ETL systems. In the context of ETL, the correlation between Big Data V characteristics and data quality was examined.

A review of the literature on Big Data from the perspective of treatment in the system of integration of data that deals with heterogeneous sources. In addition, the representation of data quality, as generally presented in other research in the field of decision-making, has been reported.

Despite the large number of studies dealing with data quality in the field of integration tools, there are still some problems to be solved, especially the implementation of Big Data in the system in terms of the practical resources of ETL systems. An analysis of the state of the art in data quality assessment revealed that sampling has not yet been considered as a solution to the expectations of data quality testing for BI systems. We also highlight the evaluation of practical factors that must be measured to ensure data quality in relation to the dimensions of the data, as presented by the research area. In particular, data quality can be applied to the design of an ETL pipeline in a marketing data-management platform.

This study addresses these issues and proposes a test framework for evaluating data quality in a data extraction, transformation, and loading system in a data warehouse. Specifically, we compared stratified sampling to achieve data quality and a normal implementation that ensured the quality of the dataset. We presented data that demonstrated the performance of stratified sampling in an environment where the normal implementation of the dataset was not applicable. We also defined the factors involved in this process to ensure that the expected runtime and memory consumption are achieved relative to the available resources.

Finally, we developed a testbed to ensure implementation of the Stratified Sampling algorithm in a more controlled domain. To simulate the processing of large amounts of Big Data, a test environment was used to ensure the sampling of data in a text file data processing application. We described the results of performance tests and

constraints to prove the use of Stratified Sampling in models of ETL production that are subject to Big Data.

BIBLIOGRAPHY

1. Udofia, E., Buduka, S., Akpabio, J., Egwu, S., Udofia, E., & Olagunju, D. (2020). Digital Transformation: After the Big Data, What Next? B Day 1 Tue, August 11, 2020. SPE Nigeria Annual International Conference and Exhibition. SPE. <https://doi.org/10.2118/203614-ms>
2. De Mauro, A., Greco, M., & Grimaldi, M. (2015). What is big data? A consensual definition and a review of key research topics. B AIP Conference Proceedings. INTERNATIONAL CONFERENCE ON INTEGRATED INFORMATION (IC-ININFO 2014): Proceedings of the 4th International Conference on Integrated Information. AIP Publishing LLC. <https://doi.org/10.1063/1.4907823>
3. Poess, M., Rabl, T., Jacobsen, H.-A., & Caufield, B. (2014). TPC-DI. B Proceedings of the VLDB Endowment (Вип. 7, Issue 13, с. 1367–1378). VLDB Endowment. <https://doi.org/10.14778/2733004.2733009>
4. Souissi, S., & BenAyed, M. (2016). GENUS: An ETL tool treating the Big Data Variety. B 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA). 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA). IEEE. <https://doi.org/10.1109/aiccsa.2016.7945615>
5. Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002). Conceptual modeling for ETL processes. B Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP - DOLAP '02. the 5th ACM international workshop. ACM Press. <https://doi.org/10.1145/583890.583893>
6. Sreemathy, J., Brindha, R., Selva Nagalakshmi, M., Suvekha, N., Karthick Ragul, N., & Praveennandha, M. (2021). Overview of ETL Tools and Talend-Data Integration. B 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE. <https://doi.org/10.1109/icaccs51430.2021.9441984>
7. Vassiliadis, P. (2009). A Survey of Extract–Transform–Load Technology. B International Journal of Data Warehousing and Mining (Вип. 5, Issue 3, с. 1–27). IGI Global. <https://doi.org/10.4018/jdwm.2009070101>

8. Khan, M. A., Uddin, M. F., & Gupta, N. (2014). Seven V's of Big Data understanding Big Data to extract value. B Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education. 2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1). IEEE. <https://doi.org/10.1109/aseezone1.2014.6820689>
9. Saeed, N., & Husamaldin, L. (2021). Big Data Characteristics (V's) in Industry. B Iraqi Journal of Industrial Research (Вип. 8, Issue 1, с. 1–9). Corporation of Research and Industrial Development. <https://doi.org/10.53523/ijoirvol8i1id52>
10. RANJAN, J. (2019). The 10 Vs of Big Data framework in the Context of 5 Industry Verticals. B PRODUCTIVITY (Вип. 59, Issue 4, с. 324–342). Printspublications Private Limited. <https://doi.org/10.32381/prod.2019.59.04.2>
11. Chardonens, T., Cudre-Mauroux, P., Grund, M., & Perroud, B. (2013). Big data analytics on high Velocity streams: A case study. B 2013 IEEE International Conference on Big Data. 2013 IEEE International Conference on Big Data. IEEE. <https://doi.org/10.1109/bigdata.2013.6691653>
12. Dayalan, M. (2018). MapReduce: Simplified Data Processing on Large Cluster. B International Journal of Research and Engineering (Вип. 5, Issue 5, с. 399–403). Marwah Infotech. <https://doi.org/10.21276/ijre.2018.5.5.4>
13. Benjelloun, S., Aissi, M. E. M. E., Loukili, Y., Lakhrissi, Y., Ali, S. E. B., Chougrad, H., & Boushaki, A. E. (2020). Big Data Processing: Batch-based processing and stream-based processing. B 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS). 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS). IEEE. <https://doi.org/10.1109/icds50568.2020.9268684>
14. Yaqoob, I., Chang, V., Gani, A., Mokhtar, S., Hashem, I. A. T., Ahmed, E., Anuar, N. B., & Khan, S. U. (2016). WITHDRAWN: Information fusion in social big data: Foundations, state-of-the-art, applications, challenges, and future research directions. B International Journal of Information Management. Elsevier BV. <https://doi.org/10.1016/j.ijinfomgt.2016.04.014>
15. Hashem, I. A. T., Anuar, N. B., Gani, A., Yaqoob, I., Xia, F., & Khan, S. U. (2016). MapReduce: Review and open challenges. B Scientometrics (Вип. 109,

- Issue 1, c. 389–422). Springer Science and Business Media LLC.
<https://doi.org/10.1007/s11192-016-1945-y>
16. Aftab Ahmed Chandio, Nikos Tziritas, & Cheng-Zhong Xu. (2015). Big-Data Processing Techniques and Their Challenges in Transport Domain. *ZTE Communications*, 13(1), 50–59. <https://doi.org/10.3969/j.issn.1673-5188.2015.01.007>
17. Taleb, I., Serhani, M. A., Bouhaddioui, C., & Dssouli, R. (2021). Big data quality framework: a holistic approach to continuous quality management. *B Journal of Big Data* (Вип. 8, Issue 1). Springer Science and Business Media LLC.
<https://doi.org/10.1186/s40537-021-00468-0>
18. Wang, R. Y., & Strong, D. M. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. *B Journal of Management Information Systems* (Вип. 12, Issue 4, c. 5–33). Informa UK Limited.
<https://doi.org/10.1080/07421222.1996.11518099>
19. Souibgui, M., Atigui, F., Zammali, S., Cherfi, S., & Yahia, S. B. (2019). Data quality in ETL process: A preliminary study. *B Procedia Computer Science* (Вип. 159, c. 676–687). Elsevier BV. <https://doi.org/10.1016/j.procs.2019.09.223>
20. B., S., M., T., & A., A. (2015). Automated ETL Testing on the Data Quality of a Data Warehouse. *B International Journal of Computer Applications* (Вип. 131, Issue 16, c. 9–16). Foundation of Computer Science.
<https://doi.org/10.5120/ijca2015907590>
21. Cai, L., & Zhu, Y. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *B Data Science Journal* (Вип. 14, Issue 0, c. 2). Ubiquity Press, Ltd. <https://doi.org/10.5334/dsj-2015-002>
22. Hellerstein, J. M. (2018). Looking back at Postgres. *B Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker* (c. 205–224). Association for Computing Machinery. <https://doi.org/10.1145/3226595.3226614>
23. Stonebraker, M., Rowe, L. A., & Hirohama, M. (1990). The implementation of POSTGRES. *B IEEE Transactions on Knowledge and Data Engineering* (Вип. 2, Issue 1, c. 125–142). Institute of Electrical and Electronics Engineers (IEEE).
<https://doi.org/10.1109/69.50912>

24. Nguyen, T.-T., Yeom, Y.-J., Kim, T., Park, D.-H., & Kim, S. (2020). Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration. *B Sensors* (Вип. 20, Issue 16, с. 4621). MDPI AG. <https://doi.org/10.3390/s20164621>
25. Cormode, G., & Duffield, N. (2014). Sampling for big data. *B Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '14: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM. <https://doi.org/10.1145/2623330.2630811>
26. Zhang, H. L., Liu, J., Li, T., Xue, Y., Xu, S., & Chen, J. (2017). Extracting sample data based on poisson distribution. *2017 International Conference on Machine Learning and Cybernetics (ICMLC).* <https://doi.org/10.1109/icmlc.2017.8108950>
27. Zhang, H. L., Zhao, Y., Pang, C., & He, J. (2020). Splitting Large Medical Data Sets Based on Normal Distribution in Cloud Environment. *B IEEE Transactions on Cloud Computing* (Вип. 8, Issue 2, с. 518–531). Institute of Electrical and Electronics Engineers (IEEE). <https://doi.org/10.1109/tcc.2015.2462361>
28. Adhianto, L., Banerjee, S., Fagan, M., Krentel, M., Marin, G., Mellor-Crummey, J., & Tallent, N. R. (2009). HPCTOOLKIT: tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, n/a-n/a. <https://doi.org/10.1002/cpe.1553>
29. Gualo, F., Rodriguez, M., Verdugo, J., Caballero, I., & Piattini, M. (2021). Data quality certification using ISO/IEC 25012: Industrial experiences. *Journal of Systems and Software*, 176, 110938. <https://doi.org/10.1016/j.jss.2021.110938>
30. International Organization for Standardization. (n.d.). *ISO/IEC 25012*. iso25000.com. Retrieved December 9, 2022, from <https://iso25000.com/index.php/en/iso-25000-standards/iso-25012>
31. (n.d.). A study on sampling techniques for data testing. *International Journal of Computer Science and Communication (IJCSC)*. http://www.csjournals.com/IJCSC/PDF3-1/Article_3.pdf
32. Zhao, X., Liang, J., & Dang, C. (2019). A stratified sampling based clustering algorithm for large-scale data. *Knowledge-Based Systems*, 163, 416–428. <https://doi.org/10.1016/j.knosys.2018.09.007>

33. Arnab, R. (2017). Stratified Sampling. *Survey Sampling Theory and Applications*, 213–256. <https://doi.org/10.1016/b978-0-12-811848-1.00007-8>
34. Lane. (n.d.). *Transaction Processing in PostgreSQL* (By postgresql). www.postgresql.org. Retrieved December 20, 2022, from <https://www.postgresql.org/files/developer/transactions.pdf>
35. Architecture Overview — Airflow Documentation. (n.d.). <https://airflow.apache.org/docs/apache-airflow/stable/concepts/overview.html>
36. Kubernetes Components. (2022, October 24). Kubernetes. <https://kubernetes.io/docs/concepts/overview/components/>
37. Nguyen, T. T., Yeom, Y. J., Kim, T., Park, D. H., & Kim, S. (2020). Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration. *Sensors*, 20(16), 4621. <https://doi.org/10.3390/s20164621>
38. Chaya Addagarrala, K., & Kinnicutt, P. (2018). Safety critical software ground rules. *International Journal of Engineering & Technology*, 7(2.28), 344. <https://doi.org/10.14419/ijet.v7i2.28.13209>