République Algérienne Démocratique et Populaire
Ministère de l 'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE AMAR TELIDJI-LAGHOUAT-

FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

Mémoire MASTER

Domaine : Mathématique et Informatique
Filière : Informatique
Option : System D'Information et de Décision

Présenté Par :
Gozim Elhocine

THEME

Integration and Fusion Techniques in Big Data

Noms du membre de jury :

Mme. F.Guibadj.	Présidente
Dr. Younes Guellouma.	Examinateur
M. Mohammed el habib Maicha.	Encadreur

$N°$ d'ordre :...../Année Universitaire 2018/2019

Contents

Dedication

I dedicate this work to my family and many friends. A special feeling of gratitude to my loving parents, whose words of encouragement and push for tenacity ring in my ears.

My Brothers Mohamed, Bilal and Slimane.

My sisters Fatima and Ahlam have never left my side and are very special.

I also dedicate this thesis to my many friends I will always appreciate all they have done.

Acknowledgement

First and Foremost praise is to ALLAH, the Almighty, the greatest of all, on whom ultimately we depend for sustenance and guidance. I would like to thank Almighty Allah for giving me opportunity, determination and strength to do my research. His continuous grace and mercy was with me throughout my life and ever more during the tenure of my research..

I would to thank my parents for their love and support throughout my life.

I would like to sincerely thank my thesis supervisor, Prof. Mohammed el habib Maicha, for his guidance and support throughout this study and specially for his confidence in me.

To all my friends, thank you for your understanding and encouragement in many moments of crisis.

الملخص

البيانات الضخمة هو مصطلح يطلق في الحالات التي يكون فيها حجم المعطيات أو البيانات كبيرا جدا وتكون فيها سرعة تدفق المعطيات كبيرة، بالاضافة الى نوع البيانات حيث تكون مختلفة من قاعدة بيانات الى أخرى فتصبح أدوات تسيير البيانات التقليدية غير فعالة.

ان دور تجميع البيانات له دور مهم في الشركات، حيث يسمح للأشخاص أو التطبيقات بتجميع كافة البيانات المتواجدة في مختلف قواعد البيانات أو الأتية من مختلف المصادر، و رؤيتها كقاعدة بيانات موحدة.

وفي ظل عصر البيانات الضخمة أصبحت تقنيات تجميع و دمج البيانات التقليدية تواجه عدة تحديات أهمها الحجم الهائل للبيانات، سرعة تدفق المعلومات، اختلاف طبيعة البيانات كالنصوص و الصور.

و تسمح الانترنت بنشر معلومات خاطئة كما يمكن للمواقع نسخ المعلومات من بعضهم مما يؤدي الى انخفاض قيمة البيانات لدى الشركات.

في السنوات الأخيرة تم نشر الكثير من التقنيات و الحلول لمواجهة التحديات التي تواجه تجميع المعلومات و دمجها في البيانات الضخمة.

في هذه المذكرة نوضح تعريف البيانات الضخمة و أهم التقنيات الجديدة المطروحة في موضوع تجميع المعلومات و دمجها مع تطبيق احدى هذه التقنيات.

الكلمات المفتاحية :

البيانات الضخمة، تقنيات تجميع المعطيات، دمج المعطيات، تجميع المعطيات في البيانات الضخمة .

Abstract

Big data is a term for large and complex datasets that traditional processing approaches are inefficient to handle them. Big data usually comes from the Internet, enterprise systems, Internet of Things, and other information systems, Data integration has become an active area for research due to increasing of information resources with the need of users and applications to integrate and fusion data from these different sources.addressing the big data integration challenge is critical to realizing the promise of Big Data.
Big data Integration differs from traditional data integration in many dimensions : volume , velocity, variety and veracity.
In this document we shall present the definition of big data and some of technologies and tools developed to handle the big data, furthermore, we explores the new solutions have been developed by the data integration community on the topics of schema mapping, record linkage and data fusion in addressing these novel challenges faced by big data integration.
finally, we have implement the map reduce based linkage method and present the used tools, algorithms and the result thrown by this method in different computational nodes.

Keywords: Big data Integration, Big Data Fusion, Data Integration, Hadoop, MapReduce.

Résumé

Le big data, désigne des ensembles de données devenus si volumineux qu'ils dépassent l'intuition et les capacités humaines d'analyse et même celles des outils informatiques classiques de gestion de base de données ou de l'information. L'intégration de données a pour objectif de permettre un accés transparent à différentes sources de données hétérogénes.
Elle donne l'illusion à l'utilisateur (ou à l'application) d'avoir accés à un systéme unique et homogéne. La diversité des sources d'information et leur hétérogénéité est une des principales difficultés rencontrées par les systéme d'intégration et de fusion de données. L'intégration du Big Data diffère de l'intégration de données traditionnelle dans de nombreuses dimensions: volume, vitesse, variété et véracité. Dans ce document, nous présenterons la définition du big data et de certaines technologies et outils développés pour traiter les big data. De plus, nous explorons les nouvelles solutions développées par la communauté d'intégration de données.
Enfin, nous avons implémenté la map reduce based linkage et présenté les outils utilisés, les algorithmes et le résultat généré par cette méthode dans différents nœuds de calcul.

Mots clés: Le Big Data, Intégration des Données, Fusion des Données, Intégration du Big data.

List of Abbreviations

HDFS	Hadoop Distributed File System
RDBMS	Relational Database Management System
No SQL	Not Only Structured Query Language
ETL	Extract transform Load
BI	Business intelligence
JDBC	Java Database Connectivity
ODBC	Open Database Connectivity
API	Application Program Interface
DAG	Directed Acyclic Graph
ACID	Atomicity, Consistency, Isolation, and Durability (database transaction properties)
UI	User Interface
BDI	Big Data Integration
DSSP	DataSpace Support Platform
HMM	Hidden Markov Model

List of Figures

List of Tables

General Introduction

Every second sees huge amounts of exponentially growing data, being generated, stored, and analyzed, the revolution in the generation of massive data amounts comes along with the Internet usage which allows data exchange between various electronic devices and humans.
companies are shifting their focus to analyzing and learning from all this large data understand their data and transform their data into practical information helps in making better predictions and smarter decisions. This phenomenon is called "Big Data" and is identified on the emerging technology hype cycle as one of the biggest IT trends of the last few years.
managing and analyzing data, often require integrating available data sources and providing a uniform interface to access data from different sources.
companies are increasingly need to integrate and fusion information from multiple data sources. in big data environment data from different sources are of different types, many of the data sources are very dynamic, the data sources are of widely differing qualities, with significant differences in the coverage, accuracy and timeliness of data provided, and traditional data integration techniques are inefficient to handle such situation.
The objective of this master thesis is to explore the domain Big Data, in particular the methods, technologies and tools developed to handle this huge amount of data , Furthermore we aim at presenting the issues related to data integration and fusion in big data environment and giving a survey on new techniques has been proposed by the researches for big Data integration and fusion. in order to put the reader in the right way, we have implement the map reduce based linkage method and present the used tools, algorithms and the result thrown by this method in different computational nodes.
finally, we put several challenges faced by the researches in the topic of big data integration and fusion.

The rest of this thesis is structured as follows.

Chapter 1: Highlights what exactly Big Data is all about and illustrate the traditional data integration and fusion techniques.

Chapter 2: Defines the challenges of data integration and fusion related to big data environment, and giving a survey on the new techniques has been proposed by the data integration community on the topics of schema alignment, record linkage and data fusion.

Chapter 3: Present the implementation of Map-Reduce based linkage method we will examine the results thrown by the method, from the point of view of the performance in time that took the execution of the method with different amounts of computational nodes.

Chapter 1

Definitions

1.1 Introduction

Today, huge data volumes are daily generated at unprecedented rate from heterogeneous sources (e.g., health, social networks, marketing, financial). This is due to many technological trends, including the Internet Of Things and the Cloud Computing, as well as the spread of smart devices. Behind the scene, powerful systems and distributed applications are supporting such multiple connections systems. Before Big Data revolution, companies could not store all their archives for long periods nor efficiently manage huge data sets. certainly, traditional technologies have limited storage capacity, rigid management tools and are expensive. they lack of scalability, flexibility and performance needed in Big Data context. as a matter of fact, Big Data management requires significant resources, new methods and powerful technologies.

More precisely, Big Data require to clean, process, analyze, secure and provide a granular access to huge evolving data sets. companies and industries are more aware that data analysis is increasingly becoming a vital factor to be competitive, to discover new insight, and to personalize services.

Because of the interesting value that can be extracted from Big Data, many actors in different countries have launched important projects. As a result of the different Big Data projects across the world, many Big Data models, frameworks and new technologies were created to provide more storage capacity, parallel processing and real-time analysis of different heterogeneous sources. also, new solutions have been developed to ensure data privacy and security. to extract knowledge from Big Data, various models, programs, softwares, hardwares and technologies have been designed and proposed. they try to ensure more accurate and reliable results for Big Data applications.

In this chapter, we shall present the definitions of big data and several of recent technologies developed for Big Data, furthermore we illustrate the traditional data integration and fusion.

1.2 Definition of Big Data

Big data is a large and complex collection of data sets, which is difficult to process using on-hand database management tools and traditional data processing applications[11].

Big Data can be also defined using the Vs:

1.Volume: refers to size of the data from Terabytes (TB) to Petabytes (PB), and related big data structures including records, transactions, files, and tables. Data volumes are expected to grow 50 times by 2020.

2.Variety: Data sources (even in the same field or in distinct) are extremely heterogeneous. The files comes in various formats and of any type, it may be structured or unstructured such as text, audio, videos, log files and more.

3.Velocity: The data comes at high speed, some organizations data velocity is main challenge. The social media messages and credit card transactions done in millisecond and data generated by this putting in to databases.

4.Value: Which addresses the need for valuation of enterprise data? It is a most important v in big data. Value is main buzz for big data because it is important for businesses.

5.Veracity: refers uncertainty of data i.e. quality of data being captured. Data like posts on social networking sites are imprecise.

1.3 Big Data Layered Architecture

the big data system could be divided into a layered framework, as represented in Figure 1.1. The big data layered architecture is constructed of three Levels.

The infrastructure layer consists of a pool of computing and a storage including cloud computer infrastructure. the infrastructure must deliver the request of big data in terms of maximizing system utilization and storage requirements.

The computing layer is a middle ware layer and contains several big data tools for data integration, data management, and the programming model.

The application layer offers interfaces by the programming models to implement

diversity of data analysis functions including statistical analyses, clustering, classification, data mining, and build various big data applications.

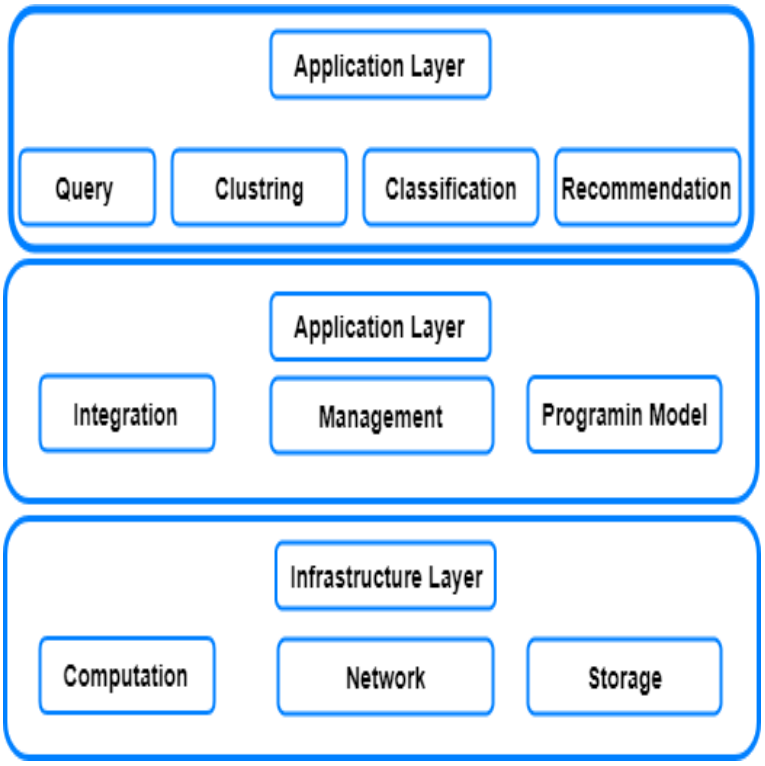

Figure 1.1: Layered architecture of big data (from[1])

1.4 Big Data Technologies

Big data is a new concept for handling loads of data thus the architectural description of this technology is modern, there are the different technologies which use almost the same approach, there are countless articles, books that describe Big Data from a technology perspective so we will attempt to present some basic principles.

1.4.1 No-SQL Database

No-SQL database is an approach to data management and data design that's useful for very large sets of distributed data. These databases are in general part of the real-time events that are detected in process deployed to inbound channels but can also be seen as an enabling technology following analytical capabilities such as relative search applications. The advantage of No-SQL is open source, Horizontal scalability, Easy to use, store complex data types, Very fast for adding new data and for simple operations/queries. The disadvantage of No-SQL is Immaturity, No indexing support, No ACID, Complex consistency models, Absence of standardization.

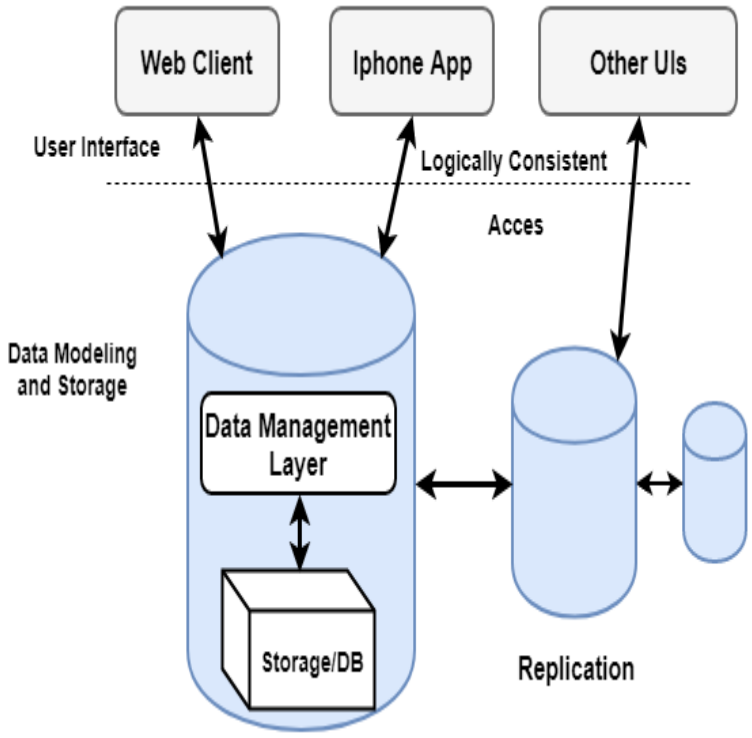

Figure 1.2: Architecture of No SQL

1.4.2 Hadoop

Hadoop was created by Doug Cutting and Mike Cafarella in 2005. Doug Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project. Formal definition of Hadoop by Apache: " The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures"[1].

[1]Apache Hadoop `https://hadoop.apache.org`

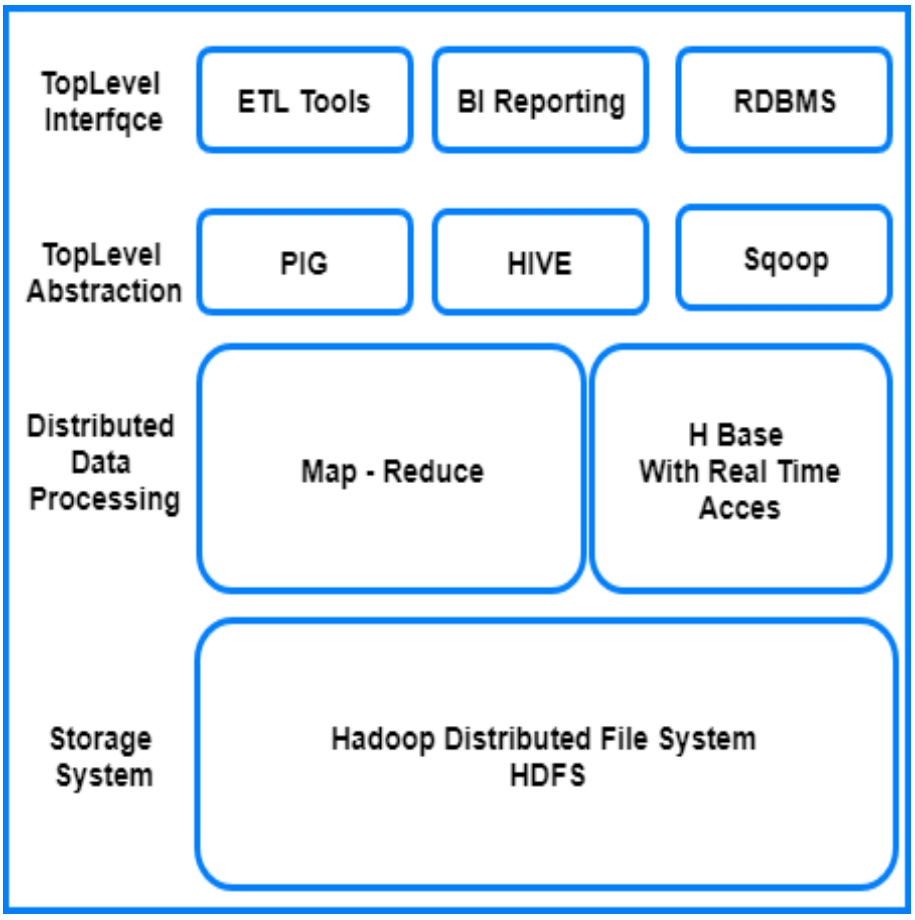

Figure 1.3: Architecture of Hadoop

A. Hadoop Distributed File System:

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject.

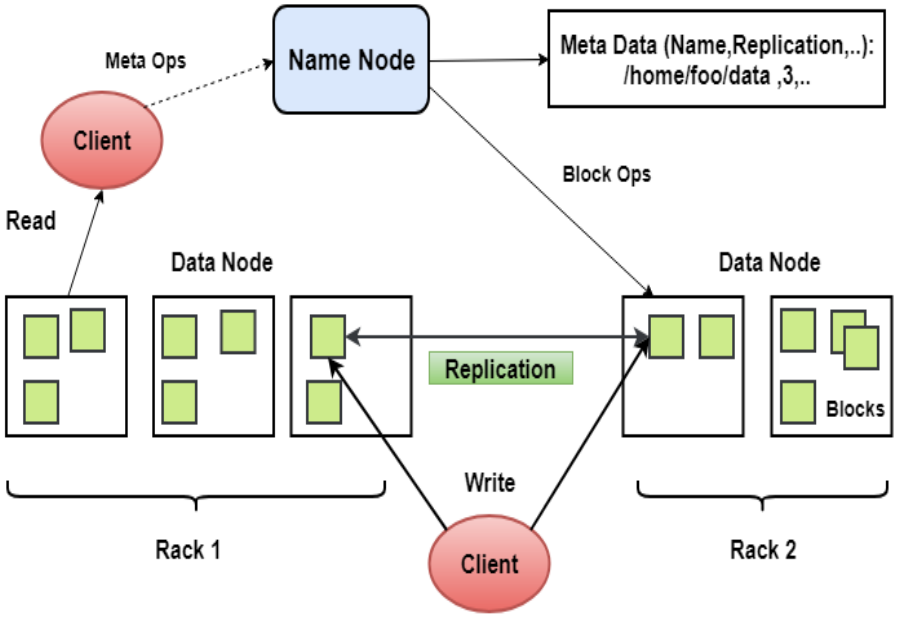

Figure 1.4: Architecture of HDFS (from [2])

- **Name Node:** manages HDFS meta data, doesn't deal with files directly.
- **Data Node:** stores blocks of HDFS—default replication level for each block: 3.

B. Map Reduce:

Map-Reduce was introduced by Google in order to process and store large data sets on commodity hardware. Map Reduce is a model for processing large-scale data records in clusters. The Map Reduce programming model is based on two functions which are map () function and reduce () function. users can simulate their own processing logic's having well defined map () and reduce () functions. Map function performs the task as the master node takes the input, divide into smaller sub modules and distribute into slave nodes.

Using Map Reduce framework the efficiency and the time to retrieve the data is quite manageable.

Map Reduce Components:

- **Job Tracker:** schedules, allocates and monitors job execution on slaves—Task Trackers.
- **Task Tracker:** runs Map Reduce operations.

C. Hive:

Apache Hive[2] is a data warehouse software for data writing and managing of large data sets in open source Hadoop platform. It converts SQL-like queries into MapReduce jobs for easy execution and processing of extremely large volumes of data.

[2]Apache Hive `https://hive.apache.org`

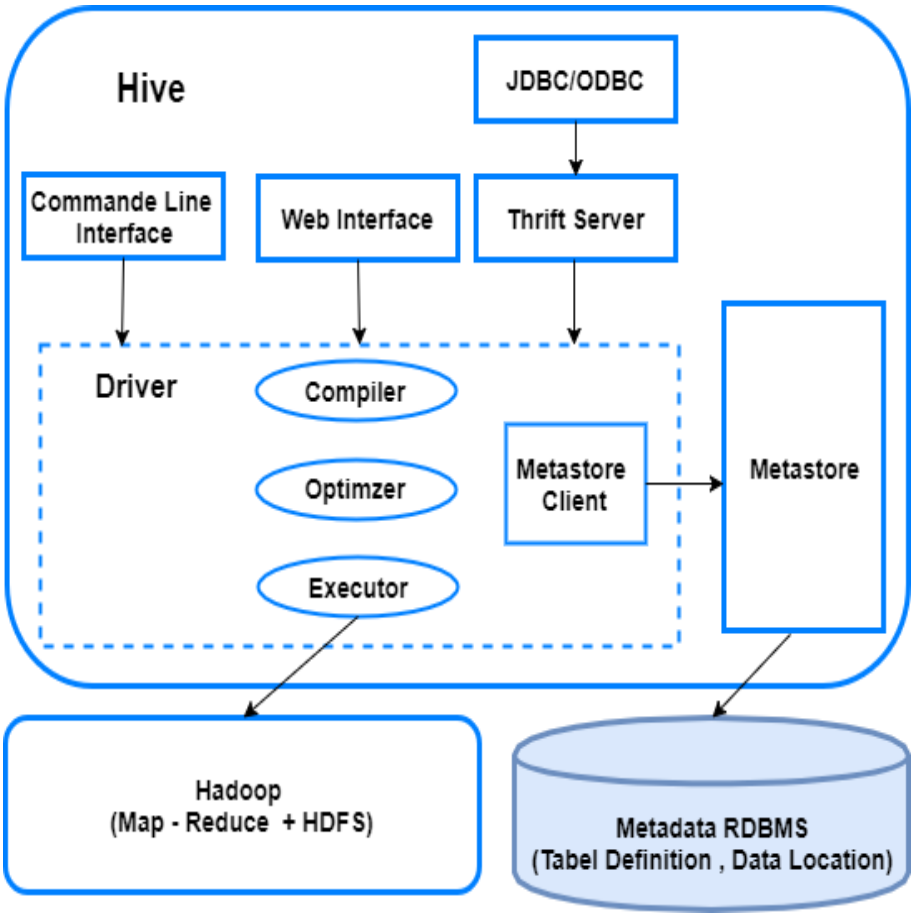

Figure 1.5: Architecture of Hive (from [3])

As shown in Figure 1.5, the main components of Hive are:

• **Driver:**The component which receives the queries. This component implements the notion of session handles and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.

• **Compiler:** The component that parses the query, does semantic analysis on the different query blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the metastore.

• **Metastore:** The component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the serializers and deserializers necessary to read and write data and the corresponding HDFS files where the data is stored

• **Executor:** The component which executes the execution plan created by the

compiler. The plan is a DAG (Directed Acyclic Graph) of stages. The execution engine manages the dependencies between these different stages of the plan and executes these stages on the appropriate system components.

Other Hadoop-related projects at Apache include:

HBase: It is open source, distributed and Non-relational database system implemented in Java. It runs above the layer of HDFS. It can serve the input and output for the Map Reduce in well-mannered structure.

Oozie: Oozie is a web-application that runs in ajava servlet. Oozie use the database to gather the information of Workflow which is a collection of actions. It manages the Hadoop jobs in a mannered way.

Sqoop: Sqoop[3] is a command-line interface application that provides platform which is used for converting data from relational databases and Hadoop or vice versa.

Avro: Avro[4] is a system that provides functionality of data serialization and service of data exchange. It is basically used in Apache Hadoop.

Chukwa: Chukwa[5] is a framework that is used for data collection and analysis to process and analyze the massive amount of logs. It is built on the upper layer of the HDFS and Map Reduce framework.

Pig: Pig is high-level platform where the Map Reduce framework is created which is used with Hadoop platform. It is a high level data processing system where the data records are analyzed that occurs in high level language.

Zookeeper: It is a centralization based service that provides distributed synchronization and provides group services along with maintenance of the configuration information and records.

1.5 Traditional data Integration

Data integration systems face two folds of challenges. First, data from disparate sources are often heterogeneous. Heterogeneity can exist at the schema level, where multiple data sources often describe the same domain using different schemas.

[3]Apache Sqoop `https://sqoop.apache.org`
[4]Apache Avro `https://avro.apache.org`
[5]Apache Chukwa `http://chukwa.apache.org`

Heterogeneity may exist at the instance level, where variety of sources can represent the same real-world entity in different ways.

Second, different sources can provide conflicting data. Conflicts can arise because of incomplete data, erroneous data, and out-of-date data. Returning incorrect data in a query result can be misleading and even harmful: one may visit a store at a wrong address, and even make bad business decisions. It is thus critical for data integration systems to resolve conflicts from several sources and identify true values from false ones.

Data integration has three broad goals: increasing the completeness, conciseness, and correctness of data that is available to users and applications. Completeness measures the amount of data, in terms of both the number of records and the number of attributes. Conciseness measures the uniqueness of object representations in the integrated data, in terms of both the number of unique objects and the number of unique attributes of the objects.

Finally, correctness measures correctness of data, that is, whether the data conform to the real world. whereas high completeness can be obtained by adding more data sources to the system. To meet these requirements, a data integration system needs to perform three levels of tasks depicted in Figure 2.1.

Figure 1.6: Three tasks in data integration (from[4]).

1.5.1 Schema Alignment

First, a data integration system needs to resolve heterogeneity at the schema level by establishing semantic mappings between contents of disparate data sources. Integrating schemas passes through several stages, namely: Schema Matching, Schema Integration and Schema mapping.

- **Schema Matching:**

Schema matching determines which concepts of one schema match those of another. if the Global Schema has already been defined, then one of these schemas is typically the Global schema, and the task is to match each Local Schema to the Global

Schema. otherwise, matching is done on two Local Schemas. The matches that are determined in this phase are then used in schema mapping to produce a set of directed mappings, which, when applied to the source schema, would map its concepts to the target schema.

• **Schema Integration:**

Once schema matching is done, the correspondences between the various Local schemas have been identified. The next step is to create the Global Schema, and this is referred to as schema integration.this step is only necessary if a Global Schema has not already been defined and matching was performed on individual Local Schema. If the Global Schema was defined up-front, then the matching step would determine correspondences between it and each of the Local Schemas and there would be no need for the integration step.

• **Schema Mapping:**

Once a Global Schema (or mediated schema) is defined, it is necessary to identify how the data from each of the local databases (source) can be mapped to Global Schema (target) while preserving semantic consistency (as defined by both the source and the target). Although schema matching has identified the correspondences between the Local Schemas and the Global Schema, it may not have identified explicitly how to obtain the global database from the local ones.[14], figure 1.7 shows the Schema integration process.

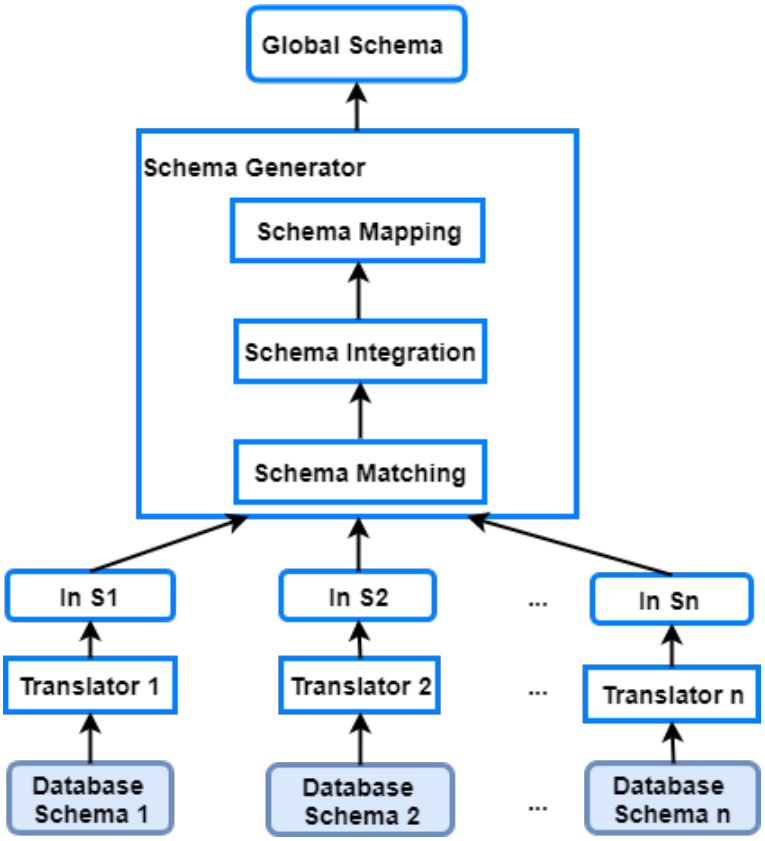

Figure 1.7: Schema Integration Process.

1.5.2 Record Linkage

Second, a data integration system needs to resolve heterogeneity at the instance level by detecting records that refer to the same real-world entity.
The Record Linkage problem, also known as Data Matching or Entity Resolution refers to identify, relate and merge records corresponding to the same entity stored in different databases. A special case arises when we analyze duplicates of the same database, this problem is known as duplicate detection. figure 1.8 shows the general process of record linkage.

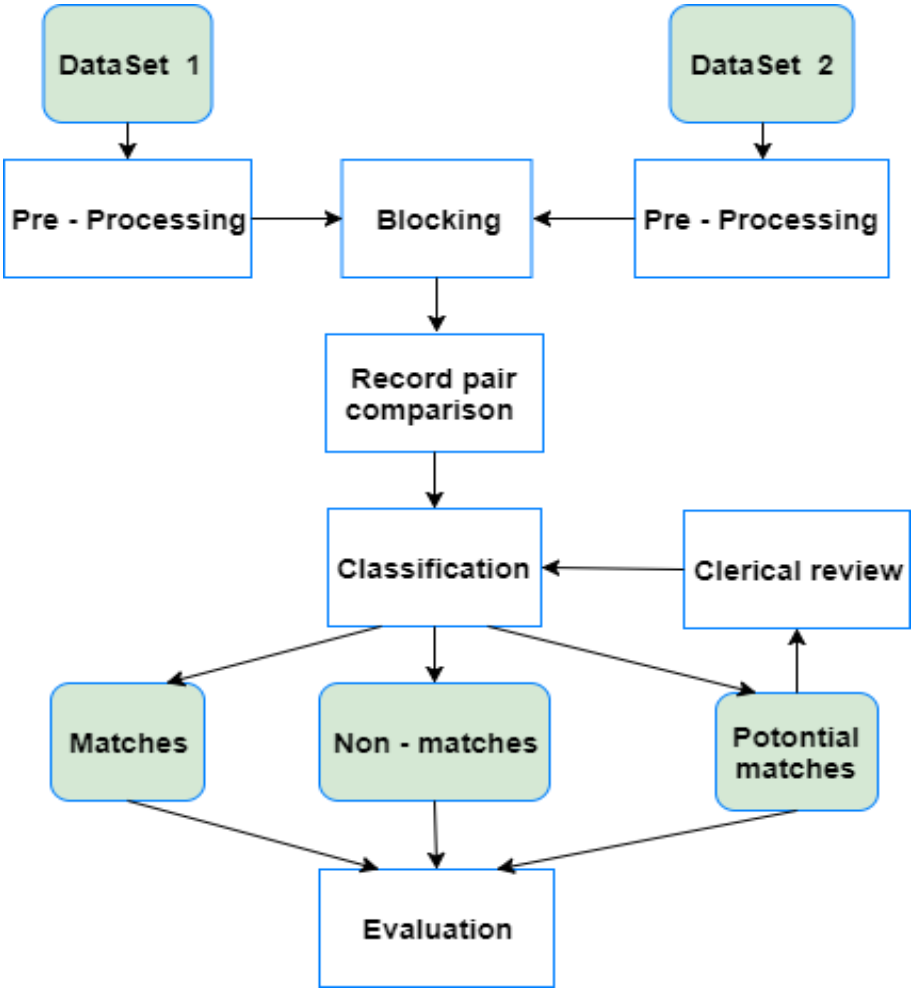

Figure 1.8: The general process of Record linkage.

A. Data Pre-Processing:

data matching commonly relies on personal information, such as names, addresses, and dates of birth, it is important to make sure that data sourced from different databases have been appropriately cleaned and standardised, the aim of this process is to ensure that the attributes used for the matching have the same structure, and their content follows the same formats.
There are three (for certain types of data possibly four) major steps involved in data

pre-preprocessing.

1. Remove unwanted characters, this step corresponds to an initial cleaning, where characters such as commas, colons, semicolons are removed.

In certain applications, some words can also be removed if it is known that they do not contain any information that is of relevance to the data matching process.

2. Expand abbreviations and correct misspellings, this second step of data preprocessing is crucial to improve the quality of the data to be matched. commonly the data matching process this step is based on look-up tables that contain name variations, nicknames, and common misspellings, and their correct or expanded versions.

3. Segment attributes into well-defined and consistent output attributes, this step deals with the common situation of database attributes that contain several pieces of information.

The process of segmenting attribute values is also called parsing, it is of high importance for both names and addresses, but also for dates.

4. Verify the correctness of attribute values, this last step can, for example, be employed for addresses if an external database is available that contains all known and valid addresses in a country or region.

B. Blocking:

A type of indexing technique that has traditional been employed in record linkage to reduce the number of record pairs that need to be compared. Blocking splits the input database(s) according to a blocking key. Only records that have the same blocking key value are inserted into the same block. Candidate record pairs are formed from all records in the same block. more details in (section 2.4.1).

C. Record pair comparison:

There are different techniques to compare the fields of the records, in general the idea that is pursued is to define a function named "sim" that complies with the following:

$$\text{sim}(\text{value 1 ,value 2 }) = 1 \Rightarrow \text{value 1 is equal to value 2.}$$

$$\text{sim}(\text{value 1 ,value 2 }) = 0 \Rightarrow \text{value 1 is completely different from value 2.}$$

$$0 < \text{sim}(\text{value 1 ,value 2 }) < 1 \Rightarrow \text{value 1 is possible similar to value2.}$$

with this definition there are many different techniques to calculate the similarity between two values depending on the type of data.

Edit Distance:

The approximation by distance in type of text fields is based on the concept of

number of operations to convert one text into another. The best known technique is Levenshtein's distance, which is precisely defined as the minimum number of insertions, substitutions and deletion of characters that must be done to transform one text into another.

To obtain the similarity function using Levenshtein's distance we must perform the following calculation:

$$\text{simlevenshtein (s1 , s2)} = 1 - \frac{distlevenshtein(s1,s2)}{max(|s1|,|s2|)}$$

In addition to these, there are numerous ways to compare the similarity of two texts, which will not go into detail in this thesis and some of them are:

- Smith-Waterman Edit Distance String Comparison.
- Q-gram Based String Comparison.
- Jaro and Winkler String Comparison.

For types of numerical data and dates there are some ways to calculate the similarity which are:

Absolute maximum value:

For certain applications, the data used in data matching not only contain string values, but also attributes that contain numerical information.

A maximum tolerance value is defined in the difference between two numbers, which is independent of its current values, leaving the similarity function as follows:

$$\text{simnum(n1 , n2)} = 1 - \left(\frac{|n1-n2|}{dmax}\right) , |n1 - n2| < dmax$$

If it is not met | n1 - n2 | <"Maximum difference" then the similarity is 0. This method can be used to compare dates, since the maximum distance can be the number of days, months, years, minutes, hours, etc. and the difference | n1 - n2 | it can be the difference in the selected measure in the maximum distance.

Percentage difference:

this technique requires entering a maximum tolerable difference percentage, so that this percentage is between 0 and 100. The similarity function is calculated as follows:

$$\text{simnumper(n1 , n2)} = 1 - \left(\frac{pc}{pcmax}\right) , \text{pc} = \frac{|n1-n2|}{max(|n1|,|n2|)} \times 100$$

If pc exceeds pc max then the similarity between n1 and n2 is 0. Like the previous method, this can also be used to compare the similarity between two dates, leaving:

$$\text{simdate(d1 , d2)} = 1.0 - \left(\frac{dpc}{dpcmax}\right) , \text{pc} = \frac{|d1-d2|}{max(|d1|,|d2|)} \times 100$$

Where | d1 - d2 | is the difference in the chosen measure between the two dates and max (| d1 |, | d2 |) is the maximum value between the two dates in the chosen

measure.

D. Record Pair Classification:

Classifying the compared record pairs based on their comparison vectors or their summed similarities is a two-class (binary) or three-class classification task. In the two-class case, each compared record pair is classified to be either a match or a non-match. The first class contains the pairs of records that are assumed to refer to the same real-world entity, while for the second class it is assumed that the two records in a pair do not refer to the same entity.

Threshold-Based Classification:

The simplest way to classify candidate record pairs into the two classes of matches and non-matches (and possibly the third class of potential matches) is to sum the similarity values in their comparison vectors into a single total similarity value (called 'SimSum', and to then apply a similarity threshold (or two in the case where potential matches are considered) to decide into which class a candidate record pair belongs.

With two classes (matches and non-matches), a single classification threshold, t, is needed for classifying a record pair(r_i, r_j):

$$\text{SimSum}[r_i, r_j] \geq t \Rightarrow [r_i, r_j] \rightarrow \text{Match},$$
$$\text{SimSum}[r_i, r_j] < t \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}.$$

With three classes (matches, non-matches and potential matches), two classifi- cation thresholds, t_l (lower) and t_u (upper), are needed, and a record pair(r_i, r_j) is classified according to:

$$\text{SimSum}[ri,rj] \geq t_u \Rightarrow [r_i, r_j] \rightarrow \text{Match},$$
$$t_l < \text{SimSum}[ri,rj] < t_u \Rightarrow [r_i, r_j] \rightarrow \text{Potential Match}.$$
$$\text{SimSum}[r_i, r_j] \leq t_l \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}.$$

The main problem with this classification method is that it does not take account that there are more important attributes than others, which carries the risk of many incorrect classifications.

To solve this problem you can enter a weight to each attribute and calculate the Sum of similarities in the following way:

$$\text{SimSumWeight}=\text{weight A1} \times \text{SimSum (R1A1,R2A1)} + ..+ \text{weight N} \times \text{SimSum (R1A}_N\text{,R2A}_N\text{)}$$

Rule-Based Classification:

It is based on applying logical rules to the similarity calculations of the different attributes. They use the form P => C, where P is the predicate and C the result of the classification. P has to be a Boolean expression and C can be Match, No Match and Potential Match. For example:

$$\text{if (simnombre (r1, r2)} > 0.7 \wedge \text{simfirstname(r1, r2)} > 0.8 \text{)} \wedge \text{simemail (r1, r2)} = 1 \Rightarrow \text{Match}$$

The rules must have the following characteristics:

1. High precision, means that a rule that classifies records in a certain class (match, potential match and non - match), has to classify the majority of those records and not classify the non-corresponding records.
2. High coverage, each rule should cover as many records as possible.

Probabilistic Classification:

It refers to the techniques developed from the 60's which are based on the idea of assigning different weights to the attributes but not only at a general level but also to local level of each record.

Cost-Based Classification:

Use rules created by Bayesian networks to determine the classification of each pair of records.

Supervised classification:

They require a set of training data, where a classification model is built, trained and evaluated, in such a way that the model can classify pairs of records whose outcome is unknown. the classification models more Popular are decision trees and support vector machine.

1.5.3 Data Fusion

Third, a data integration system needs to combine records that refer to the same real-world entity by fusing them into a single representation and resolving possible conflicts from different data sources.
there is two types of data conflicts: uncertainty and contradiction. Uncertainty is a conflict between a non null value and one or more null values that are all used to describe the same property of a real-world entity. Uncertainty is caused by missing information, such as null values in a source or a completely missing attribute in a source. Contradiction is a conflict between two or more different non-null values that are all used to describe the same property of the same entity. Contradiction is caused by different sources providing different values for the same attribute of a real-world entity[17].
Data conflicts, in the form of uncertainties or contradictions, can be resolved in numerous ways.

Conflict resolution strategies

There are many different data integration and fusion systems, each with their own solution. Fig.2.3 classifies existing strategies to approach data conflicts.

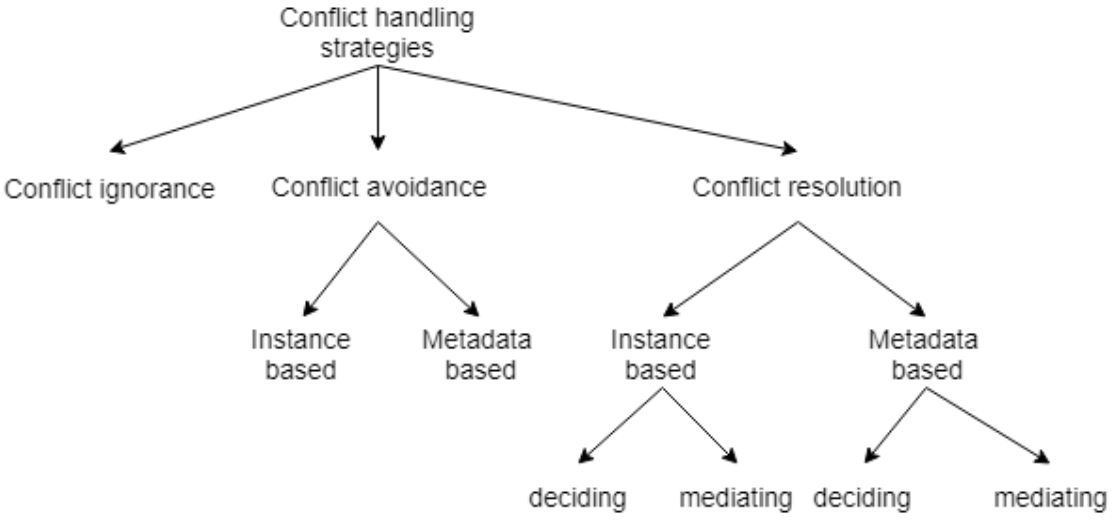

Figure 1.9: A classification of conflict resolution strategies (from [5])

In particular, Conflict ignoring strategies are not aware of conflicts, perform no resolution, and thus may produce inconsistent results. Conflict avoiding strategies are aware of conflicts but do not perform individual resolution for each conflict. Rather, a single decision is made, e.g., preference of a source, and applied to all conflicts. Finally, conflict resolving strategies provide the means for individual fusion decisions for each conflict. Such decisions can be instance-based, i.e., they regard the actual conflicting data values, or they can be metadata based, i.e., they choose values based on metadata, such as freshness of data or the reliability of a source. Finally, strategies can be classified by the result they are able to produce: deciding strategies choose a preferred value among the existing values, while mediating strategies can produce an entirely new value, such as the average of a set of conflicting numbers.Table 2.1 lists some of the strategies and their classification.

Strategy	Classification	Short Description
Pass it on	ignoring	escalates conflicts to user or application
Consider all possibilities	ignoring	creates all possible value combinations
Take the information	avoiding, instance based	prefers values over null-values
No Gossiping	avoiding, instance based	returns only consistent tuples.
Trust your friends	avoiding, instance based	takes the value of a preferred source
Cry with the wolves	resolution, instance based, deciding	takes the most often occurring value
Roll the dice	resolution, instance based, deciding	takes a random value
Meet in the middle	resolution, instance based, deciding	takes an average value
Keep up to date	resolution, instance based, deciding	takes the most recent value

Table 1.1: Conflict resolution strategies (from[8])

1.6 Database Integration Approaches

There are many ways to integrate data. the three common approaches are Data Warehouse Approach, Federated Database and Mediator approach.

1. Data Warehousing Approach

A data warehouse is a collection of data that supports decision-making processes. It provides the following features:

• It is subject-oriented.

• It is integrated and consistent.

• It shows its evolution over time and it is not volatile.

According to [14] Database integration can be either logical or physical. Data warehouse considered as physical integration because integrated data is materialized.

2. Federated Database Approach

The term " database federation " refers to an architecture in which middle-ware, consisting of a relational database management system, provides uniform access to a number of heterogeneous data sources. the data sources are federated, that is, they are linked together into a unified system by the database management system. The system shields its users from the need to know what the sources are, what hardware and software they run on, how they are accessed (via what programming interface or language), and even how the data stored in these sources are modeled and managed[19].

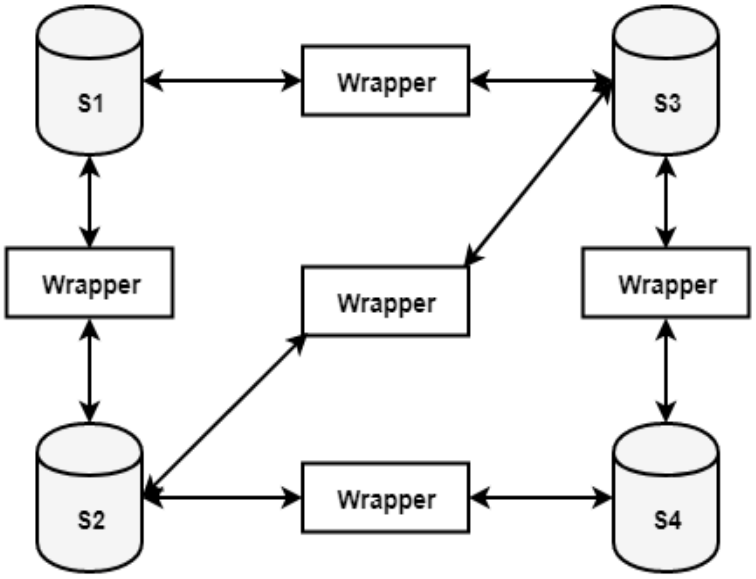

Figure 1.10: Federated Database Architecture.

3. Mediator approach

The mediator approach, originally proposed in 1992, has been used for integrating heterogeneous data in several projects. It is second generation data integration after federated framework[20].

Most mediator systems integrate data through a central mediator server accessed on or several sources through a number of wrappers interface that translate data to a common data model. However, the original goal of mediator is a distributed software module that transparently encodes domain-specific knowledge about data and share abstraction of that data with higher layers of mediators or applications[21].

1.7 Conclusion

Each second sees a huge amount of data being generated either by human collaboration or by machines which are all around us. The Age of Big Data has come and there is a need to address the challenges which come along with it. Consequently, the Big Data is widely discussed, many books and journals have been published to address its challenges, definitions and recommendations on how to deal with it. Moreover, the terms such as privacy, security and ethical problems are also considered. Although Big Data is a frequently discussed topic in theoretical manners, there is deficiency in publications and sources dedicated to its practical usage. Nevertheless, there are some evolving technologies such as Apache Hadoop along with its ecosystem. This technology is considered as the first open-source and widely used Big Data technology, building upon a distributed filesystem and an implementation of MapReduce paradigm. Most of the other technologies dedicated to deal with Big Data are based on the Hadoop solution. Although Hadoop is often discussed as a universal Big Data platform, it cannot address all Big Data problems.
Furthermore, Data integration has been an active area because it has become required in many fields. Several approaches adopted to achieve the goal of integration. No single technique is better than the others, and different integration solutions serve different purposes. there are many ways to integrate data and can be logical or physical integration.

Chapter 2

Survey on Big Data Integration and Fusion

2.1 Introduction

The term Big Data initially named from the big or large volume of data. There is no proper definition for Big Data. It was defined as a situation where the volume, velocity and variety of data exceed an organizations storage or compute capacity for accurate and timely decision making. Storage of these big data can be done by introducing multiple data centers, where as utilizing these data in an effective manner is a tedious one which is to be considered carefully.

Today, organizations invest more in data manipulation and most of the time the stored data are unused and they are not retrieved and utilized in a proper way. enterprises and organizations are spending more in data processing. actually big data is not a problem whereas it is a big asset to the organizations. Data are from different sources and integrating such data is very important and available data warehousing tools are used for doing integration. but in big data environment data from different sources are of different formats and existing data warehousing techniques in data mining are inefficient to handle such situation. this chapter present the issues related to data integration in big data environment and techniques available for Big Data Integration.

2.2 Definition

Big data integration (BDI) means linking or fusing large volumes of heterogeneous data from many dynamic data sources [10].
Big data integration differs from traditional data integration (which includes virtual integration and materialized warehousing) in many dimensions.
Volume: Not only can each data source contain a huge volume of data, but also the number of data sources,even for a single domain, has grown to be in the tens of thousands.
Velocity: As a direct consequence of the rate at which data is being collected and continuously made available, many of the data sources are very dynamic.
Variety: Data sources (even in the same domain) are extremely heterogeneous.
Veracity: The data sources are of widely differing qualities with significant differences in the coverage, accuracy and timeliness of data provided.
To addressing these novel challenges a new techniques have been proposed by the data integration community on the topics of schema mapping, record linkage and data fusion(presented in more detail below).

2.3 Techniques on Schema Alignment

the web provides huge volume of structured data, realizing the full potential of such data requires seamless integration. however, the data are at web scale, the border between the different domains is fuzzy and the variety within each domain is huge, and at every moment a new web source may appear or an existing web source may disappear. all these present big challenges for schema alignment. this section describes recent progress in schema alignment.

2.3.1 Interfaces to query the Deep Web

The Deep Web is an area of the Internet that is not indexable by search engines and not linked to pages on the Surface Web. in 2001, an initial study by Bergman indicated the size of the data in the Deep Web to be approximately 500 times the size of the data in the Surface Web which included as many as 43,000-96,000 web sites offering access to 7500 terabytes of data[22].
The Web has been rapidly "deepened" by myriad searchable databases online, where data are hidden behind query interfaces. Toward large scale integration over this "deep Web,". to help users exploring and integrating deep Web sources. a MetaQuerier has been biuld by [6], they focus on the integration of deep Web sources in the

same domain, which is itself an important integration task.

The MetaQuerier System:

The MetaQuerier system consists of two subsystems in a sequence: interface extraction and schema matching, as Figure 2.1 illustrates. the input of the MetaQuerier system is a set of Web pages (containing query interfaces) in the same domain. in practice, crawling and clustering techniques can be used to get such input.

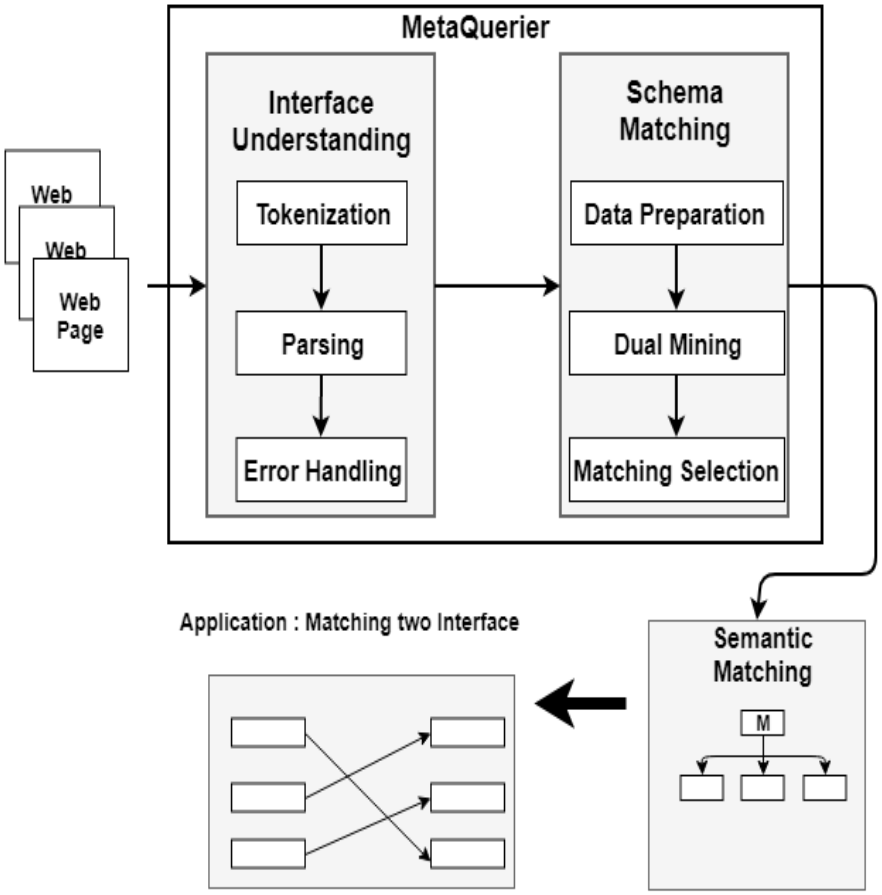

Figure 2.1: The MetaQuerier system framework(from[6]).

Interface Extraction: Interface extraction extracts the attribute information (e.g., names and domain values) from the Web query interfaces in HTML format. The Web interfaces, mostly have the some conditions patterns in layout. for instance, the most frequently used condition pattern is a text (as the attribute name) followed by an input box (as the attribute value). they pursue a parsing approach for understanding Web interfaces [23].

the interface extraction consists of:

1. tokenization: accepts Web pages as input and outputs a set of tokens as instances of terminal symbols (e.g., text, input box) in the grammar.

2. parsing: accepts tokens generated by the tokenizer and derives besteffort parse

trees based on the grammar.

3. error handling:handling conflicts and errors in the parsing result such as the situation of multiple parse trees.

Schema Matching: Schema matching discovers the semantic correspondences among the attributes in Web interfaces. for instance, in books domain, author is the synonym of the grouping of last name and first name, i.e., author = first name, last name.

Traditionally, schema matching relies on matchings between pairwise schemas before integrating multiple ones. In contrast, they proposed to exploit statistical analysis to holistically match many schemas at the same time [24]. they developed a mining approach to match schemas, consisting of:

1. data preparation: as pre-processing, data preparation accepts extracted attributes as input and outputs cleaned data by merging syntactically similar attributes.

2. dual mining: discovering matching with a dual mining of positive correlation and negative correlation.

3. matching selection: choosing the most convincing and consistent matching from the mining result.

2.3.2 Crawl and index the Deep Web data

A Web crawler is an Internet bot which helps in Web indexing. they crawl one page at a time through a website until all pages have been indexed. Web crawlers help in collecting information about a website and the links related to them, and also help in validating the HTML code and hyperlinks.

A Web crawler is also known as a Web spider, automatic indexer or simply crawler.

Traditional Web Crawler:

Traditional web crawlers are used to index the surface web. Fig.2.2 shows the working of a traditional crawler. at first, a URL is selected as the start for the web crawler. Crawler then retrieves the web pages. from the web pages, data is extracted and resource discovery is done to extract the hyperlinks, which are further processed. Data is sent to the indexer which is used as an index during search. Hyperlinks are used as the new URL and the loop continues. traditional crawler does not distinguish between pages with and without forms, structured and semi-structured data cannot be retrieved.

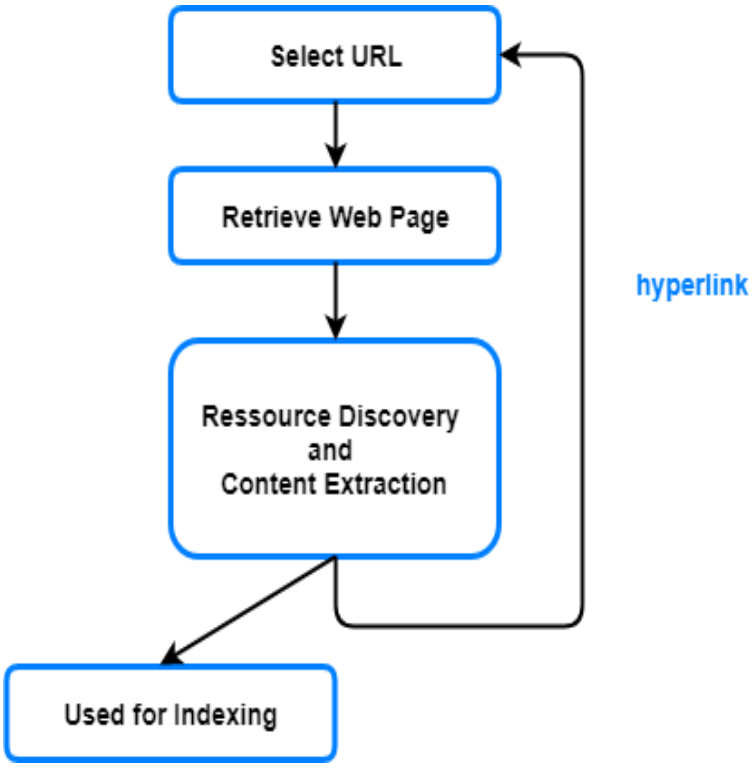

Figure 2.2: Working of traditional crawler(from[7]).

Deep Web Crawler:

Deep web data can be accessed by surfacing the web data that is not accessible to the traditional search engines. following are the major steps to access the deep web content:

Step 1: Find the Data sources.

Step 2: Data source Selection.

Step 3: Send the selected data source to the Data Integration System.

data sources for accessing the deep web may be web databases, web servers and many other dynamic web pages. depending upon the integration system, the data sources can be added. but all sources should not be included in the Integration System. the disadvantage of including all found data sources are as follows:

- Redundant data may be added.
- Irrelevant data may be added reducing the overall.
- quality of the Data Integration System.
- Low quality data can be included.
- High cost of including data since, networking and processing cost are associated with including a data source in the integration system.

Deep Web Crawlers are similar to traditional crawlers, but traditional crawlers do not distinguish between pages with and without forms. the results provided by the search engine are based on the copy of its local index. If additional steps are added to process pages, on which forms are detected and extraction of hidden information

in databases is done then the crawler is termed as Hidden/Deep Web Crawler [25]. Resource discovery and data extraction are the main task of Deep Web Crawler. Fig.2.3 shows the working of deep web crawler by addition of some extra steps. in this, the retrieved pages are checked if they have form. if form is present then it is processed to build an internal representation. forms are filled with untried values and submitted. the returned page is then analyzed to check if a valid search result was returned. if the response page contains hypertext links, these are followed and the loop continues.

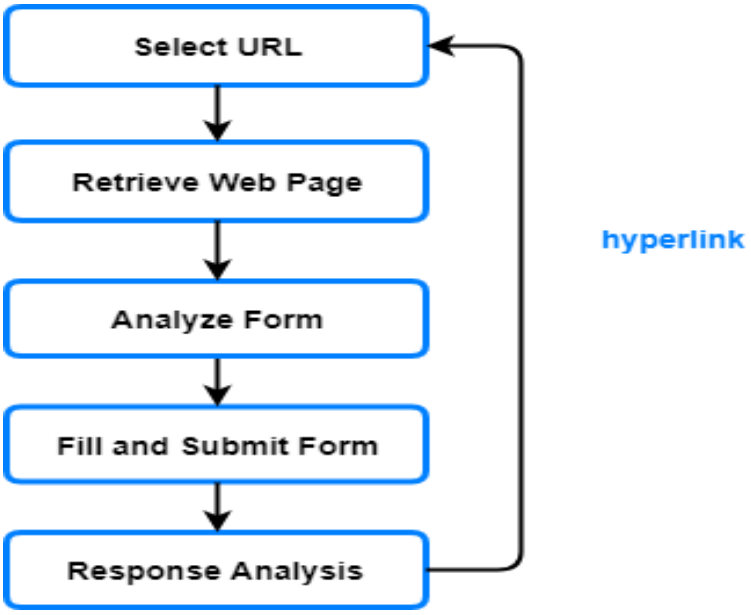

Figure 2.3: Working of Deep web crawler(from[7]).

2.3.3 Schema on Read

Schema on write has been the standard for many years in relational databases. before any data is written in the database, the structure of that data is strictly defined, and that metadata stored and tracked. the schema – the columns, rows, tables and relationships are all defined first for the specific purpose that database will serve. Then the data is filled into its pre-defined positions. the data must all be cleansed, transformed and made to fit in that structure before it can be stored [26]. Schema on read is the revolutionary concept that we do not have to know what we are going to do with our data before we store it. data of many types, sizes, shapes and structures can all be thrown into the data storage systems. when we access the data, when we query it, then we determine the structure we want to use [27]. Therefore, this approach can be called "schema on demand".
Schema on read is simple at first glance: you just write the information to the data store, unlike schema on write, which requires you to expend time before loading the

data, schema on read involves very little delay and you generally store the data at a raw level. Schema on read means you can write your data first and then figure how you want to organize it later.

Approach	Advantages	Disadvantages
Schema on write	• You know exactly where your data is. • The structure is optimized for the specific purpose. • Simple SQL queries • Fast answers. • The data quality is checked. • The data is checked against business rules. • Answers we get from querying this data are precise and trustworthy.	• The data has been altered and structured specifically to serve a specific purpose. • Query limitations. • ETL (Extract, Transform and Load) processes and validation rules take time to build, time to execute, and time to alter if you need to change it to suit a different purpose.
Schema on read	• Query capabilities are very flexible. • Different types of data generated by different sources can be stored in the same place. • This allows you to query multiple data stores and types at once.	• Since the data is not subjected to ETL and data cleansing processes, nor does it pass through any validation, that data may be riddled with missing or invalid data, duplicates... • The SQL queries tend to be very complex. They take time to write, and time to execute.

Table 2.1: Schema on read VS. Schema on write (from[9]).

2.3.4 Dataspace Systems

DBMS technology successfully resolved the physical and logical data dependence problem for structured data, but not for the highly heterogeneous data mix present in personal information. Franklin et al[28]. recognize this situation for a variety of domains, and present a broad vision for data management. they argue for a new system abstraction, a DataSpace Support Platform (DSSP), capable of managing all data of a particular organization, regardless of its format and location. Unlike standard data integration systems, a DSSP does not require expensive semantic data integration before any data services are provided.
For example, keyword searches should be supported at any time on all data (schema later or schema never)[28].

2.4 Techniques on Record Linkage

Record Linkage aims to find records in a dataset that represent the same real-world entity across many different data sources. with the evolution of Big Data, new difficulties appeared to deal mainly with the 4 Vs, therefore Record Linkage in Big Data is more challenging.
To address the volume dimension, new techniques have been proposed to enable parallel record linkage using MapReduce. these include techniques for adaptive blocking and techniques that balance load among different nodes, hen the data sources are dynamic and continuously evolving, applying record linkage from scratch for each update becomes unaffordable. to address the velocity aspect, incremental clustering techniques have been proposed, Record linkage between structured and unstructured data sources arises, e.g., when linking shopping transactions of people with tweets or blog posts about their shopping experience. to address the variety aspect, techniques have been proposed that tag and match free text to structured data.
This section present the news techniques developed to apply Record Linkage algorithms that handle the Volume,velocity, variety and the veracity properties of Big Data.

2.4.1 Adaptive Blocking

Record linkage of millions of individual records for ethically-approved research purposes is a computationally expensive task. blocking methods are used in record linkage systems to reduce the number of candidate record comparison pairs to a feasible number whilst still maintaining linkage accuracy.
there is many techniques for blocking in the literature here are some of they.

Standard Blocking:

The Standard Blocking (SB) method clusters records into blocks where they share the identical blocking key [29]. A blocking key is defined to be composed from the record attributes in each data set. An example of a blocking key is the first four characters of a surname attribute. A blocking key can also be composed of more than one attribute, for example, a postcode attribute could be combined with an age category attribute.

Sorted Neighborhood:

The Sorted Neighborhood (SN) method extends the idea of the classical blocking index in that the values of the blocks, are sorted alphabetically and then a sliding window is moved over these sorted blocks [31].
When a sorting index is initialized, one argument besides the base class arguments that needs to be given is:

• window size a positive integer that gives the size of the sliding window.

For example, lets assume there are five blocks in our index, and we have a sliding window size of 3 :

b1:[1,2,8,6]

b2:[22,18,33,54]

b3:[41,45,65]

b4:[31,72,76,81]

b5:[91,95]

While with the blocking index only the records within one block are compared with the records in the corresponding block of the second data set, with the sorting index and window size 3 larger blocks are formed by combining three consecutive blocks together:

b1,b2,b3: [1,2,6,8,18,22,33,41,45,54,65]

b2,b3,b4: [18,22,31,33,41,45,54,65,72,76,81]

b3,b4,b5: [31,41,45,65,72,76,81,91,95]

The use of the window limits the number of possible record pair comparisons for each record to 2w - 1 (where w is the size of the window). The resulting total number of record pair comparisons (assuming two data sets with n records each) of the sorted neighbourhood method is O(wn)[30].

Bigram Indexing:

This index implements a data structure based on bigrams and allows for fuzzy blocking. The basic idea is that after an index has been built, the values of the blocking variables will be converted into a list of bigrams, and permutations of sub-lists will be built using a given threshold (a number between 0 not including and 1 including) of all possible permutations [32].

For example, a blocking key value 'gozim' will result in a bigram list ('go','oz','zi','im'). With a threshold of 0.8 the following sub-lists of length 4 (calculated as the length of the bigram list times the threshold: 5×0.8) will be inserted into the inverted index:

('go','oz','zi') ('go','oz','im') ('go','zi','im') ('oz','zi','im')

So, the corresponding record number will be inserted into the inverted index blocks with keys: 'goozzi','goozim','goziim','ozzim'

2.4.2 MapReduce based linkage

In the last years there have been several studies of MapReduce techniques to compare large databases, especially in data matching.

In [33] a method is proposed to distribute the load among different computing nodes in order to solve the record linkage problem with MapReduce using standard blocking. another solution [34] extends the proposed method [33] in order to include pre-processing tasks indexed by one or two key field, and record comparison using one, two or more data sets.

Function of MapReduce:

When a program executes a MapReduce method, the following sequence of actions occurs:

1. Split input files: the data entry is divided into N files.

2. Map Phase: the mapper performs the interesting user-defined work of the first phase of the MapReduce program. given a key and a value, the map() method emits (key, value) pair(s) which are forwarded to the Reducers.

3. Partition and Shuffle: After the first map tasks have completed, the nodes may still be performing several more map tasks each. but they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers. this process of moving map outputs to the reducers is known as shuffling. A different subset of the intermediate key space is assigned to each reduce node; these subsets (known as "partitions") are the inputs to the reduce tasks. Each map task may emit (key, value) pairs to any partition; all values for the same key are always reduced together regardless of which mapper is its origin.

4. Reduce Phase : once all the mappers have finished the master process, it notifies each reducer the addresses of the intermediate files that were assigned to him to process. Each reducer will use remote processes to access this information, then sends it to the reducer process entered by the user which processes the entry of (Key, Values) and emit results.

Record Linkage with MapReduce :

the general Record linkage workflow with MapReduce is relatively straightforward by implementing pre-processing and blocking within the map function and by implementing matching and classification within the reduce function. to this end, map first determines the blocking key for each entity.

The MapReduce framework groups entities with the same blocking key to blocks and redistributes them. the reduce phase then matches the entities within one block.

2.4.3 Incremental linkage

The big data era raises challenges for record linkage. first, the volume of data and second is the velocity of data updates is often high, quickly making previous linkage results obsolete. these challenges call for an incremental strategy, such that

can quickly update linkage results when data updates arrive.
In [35] they define this problem as follow:

Problem definition

Given a set of records, record linkage is essentially a clustering problem, where each cluster contains records that correspond to a single distinct real-world entity.
denote by D a set of records and by C_D a clustering of records in D as record-linkage results.
Ideally, the clustering should have both high precision (i.e., records in the same cluster refer to the same real-world entity) and high recall (i.e., records referring to the same real-world entity belong to the same cluster). they denote by F the batch linkage method that obtains C_D on D that is:

$$F(D) = C_D.$$

they consider three types of update operations: Insert adds a new record; Delete removes an existing record; and Change modifies one or a few values of an existing record. note that Change can be achieved by first removing the old record and then inserting the new record. however, considering Change directly can be more efficient. they call those update operations (Insert, Delete, and Change) made at the same time an increment, denoted by $\triangle D$. the result of applying $\triangle D$ to D by $D + \triangle D$. Note that because $\triangle D$ can contain deletes and changes, the number of the resulting records may be lower than the sum of the number of original records and the number of records in the increment that is;

$$|D + \triangle D|) < |D| + |\triangle D|.$$

2.4.4 Linking text to structured data

Record linkage between structured and unstructured data sources arises, for instance, an e-commerce catalog typically comprises of specifications for millions of products. the search engine receives millions of sales offers from thousands of independent merchants that must be matched to the right products.
To address the variety aspect, techniques have been proposed that tag and match free text to structured data[36].
Following are the major steps in these technique:

1. Semantic Parsing: the matching algorithm is based on understanding the semantics of the offer descriptions and using these semantics to aid matching.this step consist of three stage process tagging the offer with attributes, identifying plausible parsing based on the tags, and finally obtaining an optimal parse.

2. Similarity Feature Vectors: The similarity between an offer and a product is defined in terms of their similarity on the values of the attributes present in them.

3. Matching Function: the matching function provide a probabilistic score of match between an offer and a product.
4. Online Matching: in the online matching phase applying a blocking strategy to reduce the candidate set to only those products that belong to the offer category then given a previously unseen offer and the goal is to identify the best matching product.

2.5 Techniques on Data Fusion

Unlike schema alignment and record linkage, data fusion is a new field that has emerged only recently. Its motivation is exactly the veracity of data: the Web has made it easy to publish and spread false information across multiple sources and so it is critical to separate the wheat from the chaff for presenting high quality data.
To address such veracity related challenges, techniques have been proposed to find the single truth from conflicting values, and to find multiple truths. Such techniques have also been extended to handle the volume of data (online data fusion), velocity of data (truth discovery for dynamic data), and variety of data (combining record linkage and data fusion).

2.5.1 Online Data Fusion

The internet contains a huge volume of structured data in different domains, but a lot of data are erroneous, and they can be copied from various sources. while data integration techniques allow querying structured data on the Web, they take the union of the answers retrieved from different sources and can thus return conflicting information. data fusion techniques, aim to find the true values, but are designed for offline data aggregation and can take a long time.
to address the problem of huge volume of data a new solution called ' Solaris ' have been proposed.
Solaris is the first online data fusion system. It starts with returning answers from the first probed source, and refreshes the answers as it probes more sources and applies fusion techniques on the retrieved data. for each returned answer, it shows the likelihood that the answer is correct, and stops retrieving data for it after gaining enough confidence that data from the unprocessed sources are unlikely to change the answer. here are four major components for such a system: truth finding, probability computation, termination justification, and (offline) source ordering [37].

2.5.2 Fusion in a Dynamic World

Modern information management applications often require integrating data from a variety of data sources. Among these sources, some may copy others, crawl or aggregate data from others, exchange data with or buy data from other sources. Sources may provide out-of-date and erroneous data, and such data can be propagated by copiers, resolving conflicts in data from different sources and determining the true values is critical for improving quality of integrated data.
However, the real world is dynamically changing (e.g., people's contact information changes over time, restaurants open and go out of business), and sources often frequently update their data to capture the changes.
to address the velocity dimension of big data a new technique have been proposed by [38].
they examine the update history of sources and study how to decide the evolving copying relationship between sources and the evolving true values, their first contribution is to propose several quality measures of data sources, these measures include:
coverage: how many values in the history a source cover.
exactness: how many updates conform to the reality.
freshness: how quickly a source captures a new value.
their second contribution is a set of HiddenMarkovModels (HMM) that decide whether a source copies from another source and at which moment it copies.
third, they develop a Bayesian model to decide when the true value for a particular data item changes and what the new value is.

2.5.3 Combining Fusion with Linkage

The size of information produced in the world increases by 30% every year and this rate will only go up. in many domains, there exist a large number of data sources and a lot of their data overlap, different sources can provide information about the same real-world entities. though, they may represent the same attribute value in different ways, and some may even provide erroneous values.
Traditional integration techniques handle this case in two steps: first, the record linkage step then, the data fusion.
According to [39] traditional integration techniques have at least three problems: first, erroneous values may prevent correct linking. second, such techniques can fall short when exceptions to the uniqueness constraints exist. third, locally resolving conflicts for linked records may overlook important global evidence.
they presents a novel technique to solve the record linkage problem with uniqueness

constraints and erroneous values. the key is to merge the linkage step and the fusion step, so we are able to identify incorrect values and differentiate them from alternative representations of the correct value from the beginning, and obtain better linkage results. Another crucial part is to make global decisions based on which sources associate a pair of values in the same record, so we can obtain better fusion results. finally, although their solution relies on uniqueness constraints to detect erroneous values, they allow a small number of violations to capture real world exceptions.

2.5.4 Truth Discovery

The Web has been changing our lives enormously, the amount of useful information available on the Web has been growing at a dramatic pace in recent years.
various web source provide conflict information to address the veracity dimension a new technique have been proposed by [40].
which studies how to find true facts from a large amount of conflicting information on many subjects that is provided by various web sites. they design a general framework for the Veracity problem, and invent an algorithm called TruthFinder, which utilizes the relationships between web sites and their information, i.e., a web site is trustworthy if it provides many pieces of true information, and a piece of information is likely to be true if it is provided by many trustworthy web sites.

2.6 Conclusion

Today data are being generated, collected and analyzed at an unprecedented scale which creates problems in management and manipulation of data. addressing the Big data integration and fusion challenge is critical to realizing the promise of big data, of enabling the large companies to make valuable, data-driven decisions to alter all aspects of society.
Big Data Integration and fusion raise in many dimension, including volume and number of sources, velocity, variety, and veracity.
The importance of big data integration and fusion has led to a substantial amount of research over the past few years on the topics of schema alignment, record linkage and data fusion to deal with the novel challenges faced by big data integration and fusion. we shall present some detailed, recent innovative techniques that have been proposed to address the big data integration and fusion challenges.
Table 3.1 shows a summary of these techniques.

	Schema alignment	Record linkage	Data fusion
Volume	Integrating Deep web , web table	Adaptive blocking, MapReduce-based linkage	Online fusion
Velocity		Incremental linkage	Fusion in a dynamic world
Variety	Dataspace systems	Linking text to structured data	Combining fusion with linkage
Veracity		Value-variety tolerant linkage	Truth discovery

Table 2.2: Summary of Big Data Integration Techniques (from[10]).

Chapter 3

Implementation of Map Reduce Based Linkage

This chapter present the implementation of the Map Reduce based Linkage method(section 3.4.2), this method include pre-processing task indexed by standard blocking technique, record comparison and classification.
In addition, we discuss the used data sets, tools and algorithms, finally we will examine the results thrown by the methods, from the point of view of the performance in time that took the execution of the method with different amounts of computational nodes.

3.1 Used Data Set

We use a data set of 500,000 records with information of people stored in CSV format downloaded from [41], Then the data set was exported to a text file.
we take only the following fields:

Name	Description	Data type
EmpID	employee id	integer
First Name	first name	String
Last Name	last name	String
E Mail	personal email	String
Date of Birth	date of birth	date
Phone No.	phone number	integer
County	country	String
City	city	String
Zip	postal code	integer

Table 3.1: Data Set Dictionary

3.2 Description of the Method

In this section we will explain the implementation of the method, which consists of two class, the Mapper that is executed first provide the pre-processing and blocking tasks, then Reducer that compare and classify the records lateof Map Reduce Based Linkage. **Definition of Mapper and Reducer :**

First, we define the main method that create a jobs by specifying for each Job which is the Map, Reducer, Partitioner and Combiner class, if we don't define the partition and the combiner classes hadoop use the default class, in addition for each job it must be indicated which is the data source and where to store the result of the execution.

Implementation of Mapper class:

This class receives all records to process, and for pre-processing we performed the following task:

1. Ignore the record that have missing value in key field.
2. Replace numbers in Name field by empty characters.
3. Turn email personal into lowercase.
4. Turn first name into lowercase.
5. Validate date.
6. Generate random date if the date is missing.
7. Separate the date into the format :[day] [month] [year].

Algorithm 1 Pre-processing

```
1: input: record[]
2: output: record []
3: method ClearRecords ( record [ ] )
4: Remove unwanted Characters And Number (record [last Name] )
5: date = record [Date of Birth ]
6: Split date of birth by the delimiter
7: Checking day
8: Checking month
9: Checking year
10: CheckingPhone (record [phone])
11: CheckingZip (record [Zip])
12: CheckingCountryAndCity (record [Country])
13: CheckingCountryAndCity (record [City])
```

For the Blocking task we used the first name as a key.

Algorithm 2 Class Mapper

```
1: input: file text
2: output: (text key, text value)
3: method Map ( Text key, Text value )
4: records [ ] = split the records by the delimiter
5: for i=0; i <= records.size ; i ++
6: Blocking key ← records[Firstname]
7: OtherFields = clearRecord(records)
8: EMIT ( text Blocking key, List OtherFields )
9: end Class
```

Implementation of Reducer class:

this class receive block have the same key in format (key, list of value), the reducer extract the records from the list, then compare them and classify by the call of the method ComapareRecordAndClassification.

Algorithm 3 Class Reducer

```
1: input: blocking key and list of values
2: output: Match result
3: method reducer ( text key, list of values )
4: number=0
5: for i = 0 ; i < values.size() ; i++
6: record1 = values.get(i)
7: split record1 into fields by the delimiter
8: for j = i+1 ; j < value.size ; j++
9: record2 = values.get(j)
10: split record2 into fields by the delimiter
11: compare record 1 and record 2 and classify as Match or Non Match
12: if (result = "MATCH" )
13: number ++
14: Finalresult.add (record 1)
15: end if
16: if reducer finish write the final reslut and the number of all duplication records
```

for the comparison of fields we used the following algorithms:

Field	Comparison Algorithm
Last Name	Jaro - winkler String Comparison
E Mail	Jaro - winkler String Comparison
Date of Birth	Numerical Comparison
Phone No.	Numerical Comparison
County	Jaro - winkler String Comparison
City	Jaro - winkler String Comparison
Zip	Numerical Comparison

Table 3.2: Used Algorithms for fields comparison.

Finally, for the classification of records we used the threshold based classification.

Algorithm 4 Comparison and classification of records

```
1: input: two records
2: output: result "Match" or "Non Match"
3: method CompareRecordandClassification (record1[ ] , record2[ ] )
4: compare fields and return the similarity
5: sumsim = sum of similarity of all fields in the record
6: if (sumsim > MaxThreshold) then
7: return "Match"
8: else if (sumsim < MaxThreshold and sumsim > MinThreshold) then
9: return "Potontial Match"
10: end if
11: else return "Non Match"
12: end if
```

3.3 Discussion and Results

In this section we will examine the results thrown by the method, from the point of view of the performance in time that took the execution of the method with different amounts of computational nodes.

we used Oracle VM Virtual box for creating Hadoop cluster, Figure 3.2 shown the High level diagram of the VirtualBox cluster running hadoop nodes, and the characteristics of the the host machine.

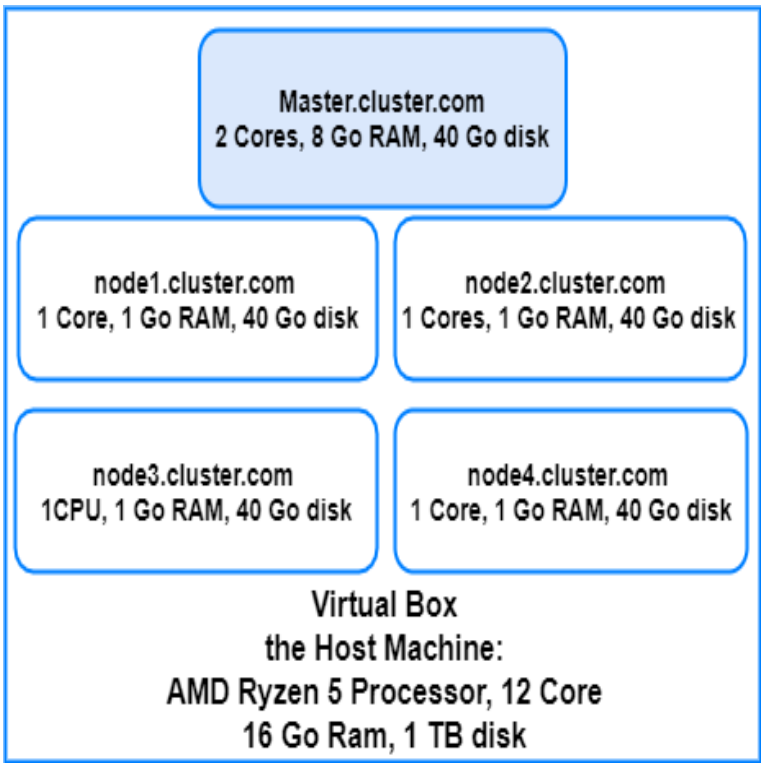

Figure 3.1: High-level diagram of the VirtualBox Cluster.

For the evaluation of the method we used the following tools:

- Cloudera Manager 5.15 [1].
- Oracle VM Virtual Box [2].
- Apache Hadoop 3.1.1.
- Java 8 for the implementation of the MapReduce.
- IDE used was Netbeans.

after the execution we have a 58 of duplication records, table 4.3 shown the time of execution with the number of nodes.

Number of Nodes	Time of Execution
1 node	3m 22.0 s
2 nodes	2m 33.037 s
4 nodes	1m 30.014 s

Table 3.3: Interpretation of the Result.

As we can see, using only one node we obtained the least performance in time, as we adding more nodes we obtain better performance.

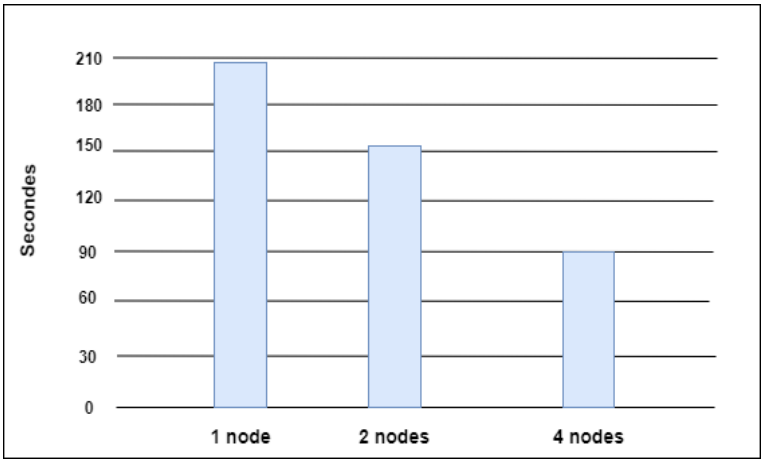

Figure 3.2: Time of execution vs Number of nodes.

Limitations of the Study:

This thesis has some limitations, because of the limitations in materials , resources, and in the used methods. the implementation of big data integration and fusion techniques required a supercomputer with a high level of performance.

[1] Cloudera `https://www.cloudera.com`
[2] Oracle VM `https://www.virtualbox.org`

Conclusion and Perspectives

In this thesis, we addressed the problem of integration and fusion in Big Data environment. one of the main contributions of our work is to explore the Big Data domain and the technologies devlopped to handle this large data sets such as Apache Hadoop along with its ecosystem, this technology is considered as the first open-source and widely used Big Data technology, building upon a distributed file system and an implementation of MapReduce paradigm, most of the other technologies dedicated to deal with Big Data are based on the Hadoop solution.

Furthermore, the problem of data integration and fusion is widely discussed, many books and journals have been published to address its challenges, definitions process and techniques.

Moreover, The importance of big data integration has led to a substantial amount of research over the past few years on the topics of schema alignment, record linkage and data fusion to deal with the novel challenges faced by big data integration.

The main focus of our thesis was on giving the outline of challenges of data integration related to Big Data environment. From the study of Big Data Integration and fusion, it is identified as the existing techniques and approaches are inefficient to handle the problems, therefore a new framework, techniques or algorithms can be proposed in future.

Finally we implement the map reduce based linkage method to touch the implantation aspect and we present the tools and algorithms used, and examine the result thrown by the method in different computational of nodes.

Bibliography

[1] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE access*, vol. 2, pp. 652–687, 2014.

[2] Hdfs architecture guide. [Online]. Available: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[3] Hive architecture. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Design

[4] X. L. Dong and D. Srivastava, "Big data integration," *Synthesis Lectures on Data Management*, vol. 7, no. 1, pp. 31–33, 2015.

[5] J. Bleiholder and F. Naumann, "Data fusion," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 1, 2009.

[6] B. He, Z. Zhang, and K.-C. Chang, "Towards building a metaquerier: Extracting and matching web query interfaces," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on.* IEEE, 2005, pp. 1098–1099.

[7] K. Khurana and M. Chandak, "Survey of techniques for deep web source selection and surfacing the hidden web content," *INT J ADV CSA*, vol. 7, no. 5, pp. 409–418, 2016.

[8] J. Bleiholder and F. Naumann, *Conflict handling strategies in an integrated information system.* Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik, 2006.

[9] S. Janković, S. Mladenović, D. Mladenović, S. Vesković, and D. Glavić, "Schema on read modeling approach as a basis of big data analytics integration in eis," *Enterprise Information Systems*, pp. 1–22, 2018.

[10] X. L. Dong and D. Srivastava, "Big data integration," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on.* IEEE, 2013, pp. 1245–1248.

[11] B. Furht and F. Villanustre, *Big data technologies and applications.* Springer, 2016.

[12] D. Calvanese and G. De Giacomo, "Data integration: A logic-based perspective," *AI magazine*, vol. 26, no. 1, p. 1, 2005.

[13] M. Lenzerini, "Data integration: A theoretical perspective," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM, 2002, pp. 233–235.

[14] M. T. Özsu and P. Valduriez, *Principles of distributed database systems.* Springer Science & Business Media, 2011.

[15] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[16] D. Dou and P. LePendu, "Ontology-based integration for relational databases," in *Proceedings of the 2006 ACM symposium on Applied computing.* ACM, 2006, pp. 461–466.

[17] X. L. Dong and F. Naumann, "Data fusion: resolving data conflicts for integration," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1654–1655, 2009.

[18] W. H. Inmon, "Building the data warehouse: getting started," 2000.

[19] L. M. Haas, E. T. Lin, and M. A. Roth, "Data integration through database federation," *IBM Systems Journal*, vol. 41, no. 4, pp. 578–596, 2002.

[20] D. George, "Understanding structural and semantic heterogeneity in the context of database schema integration," *Journal of the Department of Computing, UCLAN*, vol. 4, pp. 29–44, 2005.

[21] T. Risch, V. Josifovski, and T. Katchaounov, "Functional data integration in a distributed mediator system," in *The Functional Approach to Data Management.* Springer, 2004, pp. 211–238.

[22] M. K. Bergman, "White paper: the deep web: surfacing hidden value," *Journal of electronic publishing*, vol. 7, no. 1, 2001.

[23] Z. Zhang, B. He, and K. C.-C. Chang, "Understanding web query interfaces: Best-effort parsing with hidden syntax," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data.* ACM, 2004, pp. 107–118.

[24] B. He, K. C.-C. Chang, and J. Han, "Discovering complex matchings across web query interfaces: a correlation mining approach," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 148–157.

[25] L. Barbosa and J. Freire, "Siphoning hidden-web data through keyword-based interfaces: Retrospective," *Journal of Information and Data Management*, vol. 1, no. 1, p. 145, 2010.

[26] P. Russom, "Integrating hadoop into business intelligence and data warehousing," *TDWI Best Practices Report*, 2013.

[27] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[28] M. Franklin, A. Halevy, and D. Maier, "From databases to dataspaces: a new abstraction for information management," *ACM Sigmod Record*, vol. 34, no. 4, pp. 27–33, 2005.

[29] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989.

[30] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid, "Tailor: A record linkage toolbox," in *Proceedings 18th International Conference on Data Engineering*. IEEE, 2002, pp. 17–28.

[31] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data mining and knowledge discovery*, vol. 2, no. 1, pp. 9–37, 1998.

[32] P. Christen, T. Churches *et al.*, "Febrl-freely extensible biomedical record linkage," 2002.

[33] D. Karapiperis and V. S. Verykios, "Load-balancing the distance computations in record linkage," *ACM SIGKDD Explorations Newsletter*, vol. 17, no. 1, pp. 1–7, 2015.

[34] P. A. Albanese and J. M. Ale, "Data matching and deduplication over big data using hadoop framework," in *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016).*, 2016.

[35] A. Gruenheid, X. L. Dong, and D. Srivastava, "Incremental record linkage," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 697–708, 2014.

[36] A. Kannan, I. E. Givoni, R. Agrawal, and A. Fuxman, "Matching unstructured product offers to structured product specifications," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 404–412.

[37] X. Liu, X. L. Dong, B. C. Ooi, and D. Srivastava, "Online data fusion," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 932–943, 2011.

[38] X. L. Dong, L. Berti-Equille, and D. Srivastava, "Truth discovery and copying detection in a dynamic world," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 562–573, 2009.

[39] S. Guo, X. L. Dong, D. Srivastava, and R. Zajac, "Record linkage with uniqueness constraints and erroneous values," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 417–428, 2010.

[40] X. Yin, J. Han, and S. Y. Philip, "Truth discovery with multiple conflicting information providers on the web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796–808, 2008.

[41] Data set. [Online]. Available: http://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/

Annex

Definition of Mapper and Reducer :

```
public class MainHocine {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.err.println("Number of Parameter is not equals 2 ");
            System.exit(2);
        }
        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[1]), true);
        Job job = Job.getInstance(conf);
        job.setJarByClass(MainHocine.class);
        //set the input file and output file format
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        //set the mapper & reduce class
        job.setMapperClass(Mapper.class);
        job.setReducerClass(Reducer.class);
        //output type  of key and value for writing in file for reduce funtion
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        boolean sucess = job.waitForCompletion(true);
        if (sucess) {
            System.exit(1);
        }
    }
}
```

Figure 3.3: Definition of Mapper class and Reducer class.

Implementation of Mapper class:

```
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
    //split iput
    String[] fields = value.toString().split(",");

    StringBuilder otherFields;
    // pre - processing
    otherFields = clearRecord(fields);
    String firstName = removeUnWantedCaracterAndNumbers(fields[FIRST_NAME].toLowerCase());
    // blocking with first name
    Text field_Name_key = new Text(firstName);
    Text field_ID = new Text(fields[ID]);
    if (!firstName.equals("") && firstName.length() != 1) {
        context.write(field_Name_key, new Text(field_ID +","+ otherFields.toString()));
    }
}
public StringBuilder clearRecord(String[] record) {

    return new StringBuilder(removeUnWantedCaracterAndNumbers(record[LAST_NAME]).trim().toLowerCase() + ","
            + Methods.checkingDays(record[DATE_OF_BIRTH].split("/")[0].trim()) + ","
            + Methods.checkingMonths(record[DATE_OF_BIRTH].split("/")[1].trim()) + ","
            + Methods.checkingYears(record[DATE_OF_BIRTH].split("/")[2].trim()) + ","
            + Methods.checkingEmails(record[EMAIL].trim()) + ","
            + Methods.checkingPhone(record[PHONE].trim()) + ","
            + Methods.checkingZip(record[ZIP].trim()) + ","
            + Methods.checkingCountryAndCity(record[COUNTRY].trim()) + ","
            + Methods.checkingCountryAndCity(record[CITY].trim()));
}
```

Figure 3.4: Implementation of Mapper class.

Implementation of Reducer class:

```
// fill the records into list
public static int sum = 0;
public void reduce(Text key, Iterable<Text> value, Context context)
        throws IOException, InterruptedException {
    IntWritable in = new IntWritable();
    IntWritable su = new IntWritable();
    LinkedList<String> listOfRecodsByName = new LinkedList<>();
    for (Text registerl : value) {
        listOfRecodsByName.add(registerl.toString());
    }
    for (int i = 0; i < listOfRecodsByName.size(); i++) {
        String recordl = listOfRecodsByName.get(i);
        int index = 0;
        String[] fieldsOfRecordl = recordl.split(",");
        for (int j = i + 1; j < listOfRecodsByName.size(); j++) {
            String record2 = listOfRecodsByName.get(j);
            String[] fieldsOfRecord2 = record2.split(",");
            // fields comparison and classification
            String result = compareRegisters(fieldsOfRecordl, fieldsOfRecord2);
            // write the result into a text file
            if (result.equals("MATCH")) {
                index++;
                listOfRecodsByName.remove(j);
                j--;
            }
        }
```

Figure 3.5: Implementation of Reducer class.

Field comparison and classification functions:

```
public static String compareRegisters(String[] fieldsregister1, String[] fieldsregister2) {
        double lastNameSimi = getJaroWinklerDistance(fieldsregister1[LASTNAME], fieldsregister2[LASTNAME]);
        double dayOfBirthSimi = comparisonOfNumbers(Float.parseFloat(fieldsregister1[DAY_OF_BIRTH]),
                Float.parseFloat(fieldsregister2[DAY_OF_BIRTH]));
        double monthOfBirthSimi = comparisonOfNumbers(Float.parseFloat(fieldsregister1[MONTH_OF_BIRTH]),
                Float.parseFloat(fieldsregister2[MONTH_OF_BIRTH]));
        double yearOfBirthSimi = comparisonOfNumbers(Float.parseFloat(fieldsregister1[YEAR_OF_BIRTH]),
                Float.parseFloat(fieldsregister2[YEAR_OF_BIRTH]));
        double dateFinal=dayOfBirthSimi+monthOfBirthSimi+yearOfBirthSimi;
        if(dateFinal!=1)dateFinal=0;
        double emailSim = getJaroWinklerDistance(fieldsregister1[EMAIL], fieldsregister2[EMAIL]);
        double phoneSimi = comparisonOfStrings(fieldsregister1[PHONE], fieldsregister2[PHONE]);
        double zipSimi = comparisonOfNumbers(Integer.parseInt(fieldsregister1[ZIP]),
                Integer.parseInt(fieldsregister2[ZIP]));
        double countrySimi = getJaroWinklerDistance(fieldsregister1[COUNTRY], fieldsregister2[COUNTRY]);
        double citySimi = getJaroWinklerDistance(fieldsregister1[CITY], fieldsregister2[CITY]);
        double simsum = lastNameSimi+dateFinal + emailSim + phoneSimi + zipSimi + countrySimi + citySimi;
        float maxPercentage = 90;
        float minPercentage = 35;
        int sizeOfRecord = 7;
        float maxThreshold = (maxPercentage * sizeOfRecord) / 100f;
        float minxThreshold = (minPercentage * sizeOfRecord) / 100f;
        if (simsum >= maxThreshold) {
            return "MATCH";
        } else if (simsum < maxPercentage && simsum >= minxThreshold) {
            simsum = 2 * lastNameSimi + dayOfBirthSimi + monthOfBirthSimi + yearOfBirthSimi + emailSim +
                    phoneSimi + zipSimi + countrySimi + citySimi;
            maxThreshold = (maxPercentage * sizeOfRecord + 1) / 100f;
            if (simsum >= maxThreshold) {
                return "MATCH";
            } else {
                return "Potontial  MATCH";
            }
        }
        return "NON MATCH";
    }
```

Figure 3.6: Comparison and Classification Functions.