Journal of Theoretical and Applied Information Technology
15th September 2020. Vol.98. No 17
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SMART-ETL-MR: NOVEL ETL FRAMEWORK FOR BUILDING DATA WAREHOUSE FROM BIG DATA SOURCE USING MAPREDUCE

[1]ABDELJALIL BOUMLIK, [2]NASSIMA SOUSSI, [3]MOHAMED BAHAJ

[1 & 3] University Hassan 1, Faculty of Sciences and Technology,

Department of mathematics and computer science, Morocco

[2] National School of Applied Sciences, Sultan Moulay Slimane University, Khouribga, Morocco

E-mails: [1]boumlik.abdeljalil@gmail.com, [2]nassima.soussi@gmail.com, [3]mohamedbahaj@gmail.com

ABSTRACT

The concept of Big Data created to face the massive explosion of data produced from web 2.0, smart devices, sensors, social networks platforms like Facebook, Twitter, Instagram, LinkedIn, has increased continuously. However, new challenges and opportunities appear due to the growth of data. Nevertheless, several prominent organizations and companies have realized that data is valuable and offers competitive advantages, great benefits, and relevant knowledge when it gets converted to actionable information they can use. However, collecting these massive data is not enough, as we should be able to integrate and analyze these data pulled from different heterogeneous sources after loading them to improve analysis goals. This research article's primary objective is to adapt the ETL (extraction transformation-loading) processes with the potential of Big Data technologies in order to deal with these new challenges from data warehousing perspective and knowledge discovery that directly impacts business decision-making systems. In this article, a new approach based called SMART-ETL-MR presented on the Map-Reduce paradigm to expedite data handling and to build a well-organized data warehouse. Experimental results prove that the ETL operation performs successfully with optimal algorithms.

Keywords: *Big Data, ETL, HBase, Map Reduce, Data Warehouse, Facebook*

1. INTRODUCTION

In the last decade, we have noticed an explosion of data volume, due to the increased number of connected users to the internet via smart devices, social networks, communication systems, and heads to multiplied the size of data 300 times since 2005 till 2020. these above challenges create significant limitations on data warehouse building, exportation, processing, and knowledge discovery, etc. thus, the concept of Big Data created to face this continuous demand.

The Big Data concept refers to massive, unstructured, heterogenous datasets that cannot be handled by traditional relational database systems to maintain, capture, and manipulate data. Hence this explosion of data attains significant challenges for multiple domains in terms of decision-making systems, data mining, and knowledge discovery activities. The Big data concept requires a specific approach defined by the three V rules [18] that refer to Volume, Variety, and Velocity. Volume determines the amount of data generated in different platforms, companies, or final users, Variety involves the proliferation of heterogeneous data, and Velocity defines the frequency of generating, capturing, and sharing data.

To support this immense evolution of data and improve analysis purposes, we need to consider the essential elements of a data warehouse, called Extract-Transform-Load (ETL), typically holding up the 80% of the DW projects. The ETL process is a set of multiple operations, such as an Extraction operation covering all tasks to collect the required data. In contrast, Transformation operation consists of executing multiple series of procedures to transform the extracted data into a standard format, finally Loading operation that deals with the transformation data and load them into our Datawarehouse (DW). However, standard operations are not suitable to deal with the massive evolution of data anymore, same as the Relational Database Management Systems (RDBMS)that are not fitting for distributed databases.

For the above limitation purpose, the ETL process needs serious attention to be improved to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

deal with the massive explosion of data to prepare and handle it into the DW. Certain technologies have appeared with the emergence of Big data fields, focusing on data management such as Map-Reduce, which can deal with massive amounts of data, NoSQL databases [20] that can store unstructured data on column-oriented databases or document-oriented databases format such as MongoDB.

In this work, we present a new approach called SMART-ETL-MR that applies new breeds of big data technologies to supports scalability and performance into the Extraction and Transformation phases of integration processes like ETL. Within this context, the improvement and adaptation of ETL starts with data processing itself, in which we integrate the parallelism processing aspect by using the MapReduce paradigm for handling unstructured data to widely minimize time-consumption during this stage. In addition, the HBase database is considered in our approach as a NoSQL column-oriented data store to support complex data instead of classical Relational database or plat files (CSV, XML, etc.) that cannot deal with a massive amount of data. Moreover, The primary goal of this research is to readapt modeling ETL operations in the formal level of extraction and transformation phases and the retaining the specificities of the multidimensional structure of DW to support online analytical processing and business intelligence application by supporting the most-used operations for treating and filtering data such as Select and Join.

The remainder of this paper arranged as follows: Section II presents the most recent related works with their case studies and highlights their main limitations. Section III highlights the main concepts used in the proposed solution, such as the MapReduce paradigm, HBase database, and Data warehousing. Section IV presents the functional architecture of the conceived smart ETL with detailed algorithms for each phase. Section V illustrate a case study. Section VI represents an analysis and discussion. Finally, section VIII concludes our work and suggests some future extensions of this topic.

2. RELATED WORKS

The data warehouses are encapsulated data storage systems that allow a company or organization to combine data from various applications and sources into one single platform, more especially designed for decision support systems, data mining, and analytics reporting activities. The ETL processes gained extensive attention over the last ten years to bypass many limitations that have improved and provide benefits for data warehousing performance. In this section, we present some existing studies that deal with data integration and consider ETL modelling as an efficient solution to achieve data warehousing projects. Nevertheless, some challenges remain unsolved always or worked around. They were mainly related to the proposed algorithms, data management, handling the unstructured data, etc.

In our context of Big Data Warehousing, several approaches have been developed as part of the analysis of this massive data via the generation of OLAP Cube. The authors in [15] present, an aggregation operator called C-cube that allows data cubes to compute using column-oriented data warehouses based on the invisible join principle. Also, the work [13] describes how we can benefit from cloud computing technologies to build OLAP cubes by using Big Table and MapReduce to deploy cubes in ad-hoc Data Marts. The investigation in [14] refers to the OLAP query processing over column-oriented databases using the MapReduce framework. However, all the previous approaches, including the work presented in [12], use HBase as a very performed NoSQL database to improve OLAP querying in the Big Data Warehouse context, but they do not present any consideration of data movement from HBase that required an advanced ETL processing to be adapted with the 5Vs of big data.

Previous works have shown the potential and the advantages in time consumption during extraction, filtering, and storing large sizes of data such as P-ETL [16] and ETLMR [19], that they employed parallelism strategies with CDC functionality while integrating data. At the same time, they ignore the multidimensional aspect, which is a mandatory structure for further analysis operation. Furthermore, some other studies deal with functionalities and basic operations processing such as Select, Project, etc. using MapReduce paradigm in order to adapt classical SQL queries on Bigdata contexts as we have seen with Jack Hare Framework [17], Hadoop++ [21] and the detailed study described in [22] that also ensure the achievement of MapReduce with a set of entirely known joins strategies. The above-presented solutions are beneficial, but only for basic ETL processes which are not yet able to face the enormous amount of data.

Within the same context, the below paragraphs describe specifically the most recent paperwork's published between 2016 till 2020 and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

related to ETL approaches and highlighting their advantages and limitations.

The paper [16] proposes a platform called parallel-ETL, which is composed of five steps (extraction, partitioning, transformation, reducing, and loading), and this refers to the proposed ETL architecture. The MapReduce job commences when data sources entirely loaded via P-ETL in the HDFS, and this option was able to accelerate the ETL processing by 33%.

The paper [9] propose a novel approach called BigDimETL that start loading data from a JSON file into a CSV format, then distribute them vertically by making a correspondence between XML schema and data loaded, Aldo during transformation phase they only covered the essential operation of ETL such as Select that used to filter data.

In the same context, the author of paper [11] provides multiple parallelization and distribution of data based on each part of the ETL treatment phase (ETL processes, ETL functionalities, ETL elementary functionalities) taking into consideration the horizontal and vertical distribution.

The authors of work [6] and 2020[X]was enhancing his previous approach and covered the cases ignored in his first paperwork like, during the extraction phase, they transform JSON files directly to HBase and group them to column families after a partitioning process. From the other side, the same authors cover correctly more advanced ETL operations such as Select, project, and join operations using a shared key. Also, he creates temporary storage called DSA (Data Storage Area) to help execute the above operations as quickly as possible.

This article [7] suggests an architecture based on three levels representing the workflow of extensible ETL that deal with Big data challenges. These include the first layer called workflow designer how is communicating directly with the intermediate extensible layer which is also composed of four elements UDFs, Recommender component, a cost model, and a Monitoring agent. the primary goal of this work consisted of executing the load in a parallel manner to enhance the performance.

In article [2] proposes a novel approach called D_ELT, in which transformation of data was not taken into consideration, while the approach consists of using data already loaded on Hadoop initially. However, they tried to use a single MapReduce algorithm to handle in parallel transformation and analysis phases to use the analyzed data once transformed immediately. Nevertheless, this approach will not be efficient in handling massive data as they are transforming data before it finishes loading, and this will have a negative impact in the case of massive data.

The paperwork [4] proposed an architecture based on cloud, in which the extraction and transformation phases executed in demand with Spark. This option offers advanced and sophisticated algorithms for ETL to use, such as aggregation operations and heuristic algorithms that share results to the HDFS.

In brief, we believe that the current research works shared some similarities with ours since they are interested in building a data warehouse based on the ETL process. However, the below Table 1 present in detail that most approaches deal with basic operations using the multidimensional structure, but they did not address the Big Data context. Others handled data context and neglected the essential ETL operation. Also, the partitioning, operations processing of ETL ignored from multiple approaches. The below table illustrates the most recent papers, and if they cover the essential ETL operations.

Table 1: ETL operation covered in recent papers

Paper	Benefits	Limitations
[9]	The parallelism aspect was integrated using MapReduce, which was a benefit. Reduce the partitioning process using direct indexation.	The approach was limited as it treats only the Select operation. The Loading phase was not detailed.
[10]	The proposed approach sounds good from a performance perspective, especially during Join operation. The same approach can be adapted for addition operation such as union, and aggregation.	In this paper, the author ignores the ETL functionalities completely. The author also did not provide any algorithm related to the MapReduce process.
[6]	The author proposes an innovative that execute in parallel both transformation and analysis.	They are not truly designed to support the context of Big data. The approach does not make any

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

		cleanup of data before loading, which implies that many useless data will be loaded.
[1]	The author uses the advanced features provided by on-demand Spark during the extraction and transformation phases. we have seen an approach that combines Big data technologies with traditional technologies	The authors did not apply their approach to reel cases to demonstrate the efficiency of the solution.
[2]	The authors of this paperwork present a useful automated approach that will help developers to optimize ETL tasks.	The authors of the work ignore or neglect to present the phases clearly, and leaving it in the abstract world only, which lead to confusion and uncertainty
[3]	The MapReduce operation commences once data get loaded on the HDFS via the P-ETL. Hence, we gain time consumption during load and analysis at the same time.	Whenever the analysis operation gets repeated the time consumption will be impacted negatively (103 seconds each time) the absence of transformation before loading causes a problem of storage of large amounts of data which may not be valid, because the transformation task is associated each time with an analysis task is executed, this stored data is read repeatedly, which leads to unnecessary input/output.
[4],[5]	The author treats the ETL functionalities uniquely and neglects all additional features provided by the ETL.	The proposed approach based on Cloud, and the was limited to the use of Spark and Amazon AWS; hence he creates a limitation to other users

In fact, these previous studies are very interesting in terms of data warehousing. Although, these researches share some similarities with our solution presented in this paper, however the partitioning and selecting data vertically according to a multidimensional structure was not covered, also we focused more on reducing side-join operations which was not treated before, while it's one of the most important functionalities used during for processing large data sets.

As a conclusion, the goal of this paper is to reduce as much as possible the cost of Data warehouse implementation and produce relevant information for decision-makers and online Business Intelligent applications based on MapReduce paradigm that proves as a powerful solution for parallel processing of massive data. Indeed, the paper provides a functional architecture describing the different steps adopted by our approach to building a data warehouse from a column-oriented NoSQL database like HBase to improve OLAP querying in the data warehouse and improve the extraction and the transformation phases according to the multidimensional structure.

3. BACKGROUND

In this section, we introduce the main concepts used by our solution presented in Figure 1, that describes the proposed smart ETL process, including the MapReduce paradigm, to build a data warehouse from Twitter and Facebook as a case study based on the distributed storage HBase [6] as a NoSQL Column Oriented database [3].

3.1 HBase: NoSQL Column Oriented Database

First, The Column Oriented Database has a structure dedicated to accommodating many columns (up to several million) for each line. It characterized by a variable number of columns that can change from one row to another (we can consider that a column exists if it contains a value). At first blush, the Column-Oriented Database looks remarkably like a relational database, but the concept is entirely different.

The model of column-oriented database, as illustrated in Figure 1, is composed of a set of tables S = {T1, T2, …, Tz}; each table contains in its turn a set of rows Tk = {R1, R2, ..., Rn} with k ∈ [1, z]. Each row can be represented as Ri = (IDi, (CFi1, CFi2…, CFim)) with i∈ [1, n], IDi is a row id and CFij is a column family j ∈ [1, m] of the row Ri. Each column family can contain numerous columns that have the same categories of attributes which also called Column Qualifier, so a column

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

family CFij={CQ ij1, CQ ij2, …, CQijp} = {(Cij1, vij1), (Cij2, vij2), …, (Cijp, vijp)}.

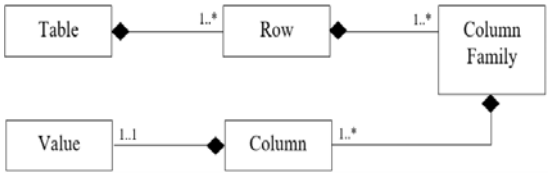

Fig. 1. Metamodel Of Column-Oriented Stores

In this approach, we have opted for HBase as a NoSQL column-oriented database used by the two famous social media systems: Facebook and twitter, representing the inputs of our system to build the targeted Data Warehouse. HBase has several advantages, such as it is high performed to improve OLAP querying in the Data Warehouse context through the multidimensional structure [9]. Besides, it characterized by high processing speed; In fact, operations such as data reading and processing will take a small amount of time as compared to traditional relational models. Therefore, the previous advantages will contribute to ensuring the proper functioning of our smart ETL.

3.2 Data warehouse

The Business Intelligence (BI) defined as a technological process that analyzes data in order to extract useful business information exploitable by leaders, managers, and other users, enabling them to make better decisions. These analytical details stored into a specific database named Data Warehouse (DW) dedicated to decision support.

The multidimensional structure of a DW defined as a star schema (S) inter-connected forming the basic structure named constellation C = {S1, S2,…, Sn}. The star schema is composed of three major elements: Fact (F) represented as a table of analysis subject (such as sale, stock …), Dimensions (D) linked to Facts in order to define the axes of analysis (such as customer, product, geography) and a Star Function (StarFct) that associates each Fact Fi∈F to a set of Dimensions. Thus, the formal structure of star schema is defined as follow: S = {F, D, StarFct} where F = {F1, F2, …, Fm} and D ={D1,D2, …, Dp}.

Each dimension denoted Di∈D with i∈[1, p], is defined by Di = {Dn, Dattr}, where Dn is the dimension name, and Dattr is a set of dimension attributes Dattr={att1,att2,…,attK} that define the level of the detail of our analysis. Each Fact denoted Fj∈F with j∈[1, m], is defined as Fj = {Fn, Fm} where Fn is the fact name and Fm is a set of measures related to Dimensions through foreign key Fm = {m1, m2, …, mY} where my = {Mn, Mt, Mf} with y ∈ [1, Y], Mn is the measure name, Mt is the measure type, and Mf is a set of aggregation functions such as AVG, MAX, MIN, and others. The metamodel in the Figure 2 illustrate the multidimensional conceptual structure of a Data Warehouse.

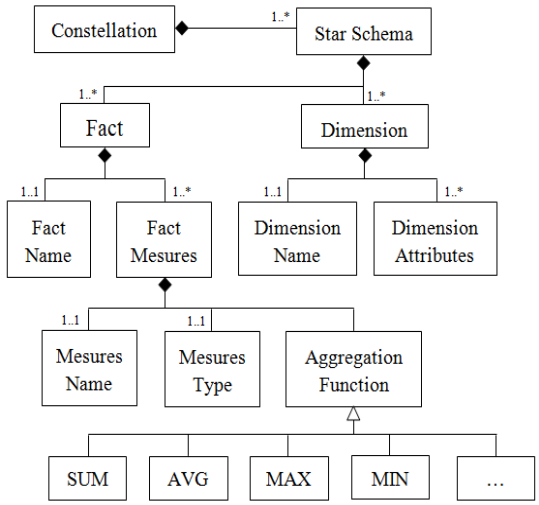

Fig. 2 Metamodel of Multidimensional Structure of DW

The regular loading of data into a Data Warehouse is ensured by ETL (Extract Transform Load); the same is a special process dedicated to managing all stages of data collection and preparation according to three principal phases:

- **Extract**: responsible for accessing heterogeneous data sources (NoSQL, triple stores, ERP...) to retrieve data, and convert it into one consolidated data warehouse format which is ready for transformation processing.
- **Transform**: to ensure better exploitation of extracted data, this phase al-lows verifying, reformatting, cleaning up, filtering and giving special treatment to missing data. Then it proceeds to apply the necessary aggregations that fuse several data to obtain a single usable and exploitable information (average, sum).
- **Load**: consist of inserting the data processed in the Data Warehouse to put them at the disposal of different tools of analysis and presentation such as Data Mining, OLAP multi-dimensional analysis, geographic analysis, decision-making, and others reporting and of course the dashboards.

3.3 MapReduce Paradigm

MapReduce is a programming model for parallel and distributed processing of massive data sets. It runs in parallel on several large clusters to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

distribute data and collect the required results. The MapReduce program is composed of the following three main functions:

- Map: takes as input a Dataset in order to split them into a smaller sub dataset, and then it carries out the required treatment on each Sub-Dataset in parallel. The output of this method is presented as a set of {key, value} pairs in which the same key can be repeated. The signature of the Map function is defined as below:
  - Map(key-in,Val-in) = {(key1, value1), (key2, value2), …, (key-n, value-n)}
- Shuffle: defined as the intermediate step between Map and Reduce functions. It operates on the outputs resulted from the different executions of Map function in order to group all {key, value} pairs having the same keys and return them as a sorted list of each key associated with their set of values. The signature of the Shuffle function presented as follow:
  - Shuffle((key1, value1), …, (key-n, value-n)) = {(key1, {value1, …, value-p}), …, (key-m, {value1, …, value-b})} with [1, p] ∪ … ∪ [1, b] = [1, n] and [1, m] ⊂ [1, n]
- Reduce: is the last step in MapReduce process, in which we reduce the previously sorted list, generated as output from the Shuffle function, into a smaller set of values sharing the same key as defined in its signature bel-low:
  - Reduce ((key1, {value1, …, value-p}), …, (key-m, {value-1, …, value-b})) = {(key-1, value-out_1), (key-2, value-out_2), …, (key-m, value-out_m)}

4. THE PROPOSED APPROACH

In this section, we describe our significant contribution summarized via the functional architecture presented in Figure 3.

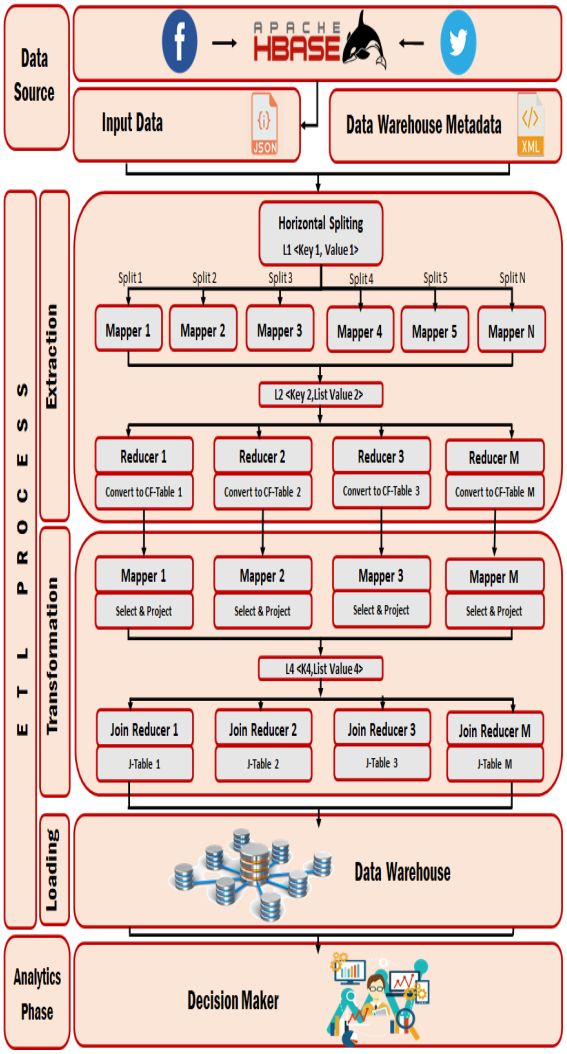

Fig.3 Global Architecture of our Approach

The above architecture is composed of three principal layers; the first one contains the column-oriented NoSQL database HBase, which is considered a data source of our system, alimented from social media (Facebook and Twitter). The second layer presents the main interfering actors that select useful and pertinent data according to the decision purpose to send it to the third layer comprising the ETL process that operates with the MapReduce paradigm as described in the subsection subsequently.

4.1 Extraction phase

The Extraction phase is the first significant step in the ETL process, aiming to prepare, clean, and convert selected data from the input sources. In our example, we collect input data from the most prominent and influential social media systems in the world: Twitter and Facebook. They provide us

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

with a vast source of knowledge discovery in various domains like decision making, marketing, needs creation strategies, and political domain. This immense data will be analyzed to select valuable analytics data to understand the users' behavior, interests, opinions, needs, and satisfaction, based on their tweets, publications, comments, pages followed, etc. However, the processing of this massive data represents severe concerns and a challenging task for professional experts and decision-makers to extract valuable analytics they need.

The input data initially formatted as a JavaScript Object Notation (JSON) file [8], which is a complex structure and cannot be considered as the best option to manipulate directly in the ETL process. Therefore, our idea consists of operating on this JSON file that contains data itself and the metadata of our Data Warehouse (DW), then focus on preparing the multidimensional schema using a parallel processing technique called MapReduce (MR) to enhance time consumption and minimize the overload of data treatment and integration by adding parallelize aspects to ETL processes. However, many related works neglect entirely to explain the implementation of MapReduce on their solutions, not even providing a realistic MR paradigm concept, which generates discrepancies in their results and presents a leak of information. Taking into consideration the above facts, we decide to demonstrate our proposed step-by-step algorithms, and we start the two key steps, Splitting and converting massive data sets into a column-oriented structure.

- **Step 1: Splitting procedure**

The splitting procedure presented in Algorithm 1 is the first step in the MR paradigm, in which we manipulate efficiently the input data represented as a tree of simple and complex objects in JSON format. We split it horizontally according to our DW metadata global structure by grouping the parsed JSON objects Oi that match the same axes of analysis Dimension (Di), or Fact (F) in one sub JSON file. Hence the output of this procedure is a list L1 that contains a set of sub JSON files built previously and represented as <Key1, Value1> in list L1, while Key1 represents their labels that corresponds to a specific Dimension (Di) or a Fact (F), and Value1 represent the corresponding nested elements.

Algorithm 1: Splitting algorithm
1: **Input**: JSON file, DW XML Metadata
2: **Output**: List L1<Key1, Value1>
3: **Begin**
4: M ← ParseXMLMetadata(metadata.xml)
5: D ← ExtractDimensions(M)
6: F ← ExtractFacts(M)
7: J ← ParseJsonFile(Input.json)
8: ElmObjects ← ExtractElementaryObjects(J)
9: **Foreach** ElmObj in ElmObjects **Do**
10: **Foreach** D and F **Do**
11: **If** ((ElmObj in D) OR (ElmObj in F)) **Then**
12: Key1 ← getLabel(ElmObj)
13: Value1← getNestedValue (ElmObj)
14: **End If**
15: **End Foreach**
16: **End Foreach**
17: **Return** L1
18: **End Algorithm**

- **Step 2: Converting procedure**

The conversion procedure triggered once the Splitting process finished successfully and consisted of converting each specific object-oriented Oi (sub-JSON files in previous L1 list) into a column-oriented table represented as a unique Column Family (CF). The Algorithm 2 illustrates the fact that each job j executes one Map function that operates itself on a specific object Oi expressed as {Key1, Value1} = {Oi.label, Oi.nestedElements}. In fact, the procedure will extract the complete set of attributes and values of each elementary object Oi and construct the output list L2, as mentioned below:

- ✓ **Key2**: receives the different combinations of attributes for each object Oi.
- ✓ **Value2**: receives the corresponding key2's values list for each instance of Key2.

In the last part of our Algorithm 2 we index the list L2 to define a targeted column-oriented table (CF-Table) that will receives data stored in the L2 list (i.e. each instance of L2 list correspond of Di or F representing the Column family label CF).

Algorithm 2: Mapping algorithm
1: **Input**: Key1(JsonLabel), Value1(Nested elements)
2: **Output**: list L2<Key2, List Value2 >
3: **Variables**: List Children, K, V, ElmtObjects
4: **String** k, v, ElmObject, Child
5: **Begin**
6: ElmtObjects ←ExtractElmryJsonObj(value1)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7: **Foreach** ElmObject in ElmtObjects **Do**
8: Children ← ElmtObject.getChildren()
9: **Foreach** Child in Children D**o**
10: k ← Child.getAttribute()
11: v ←Child.getValue()
12: K.add(k)
13: V.add(v)
14: **End Foreach**
15: **If** K in Key2 **Then**
16: getKey2(K).getValue().add(V)
17: **Else**
18: Key2.add(K)
19: Value2.add(V)
20: **End If**
21: **End Foreach**
22: **Indexing** L2 with Key1 (corresponds to targeted Di or F)
23: **Return** indexed L2
24: **End Algorithm**

The last step we have currently consists to build the column-oriented tables CF-Tables from data produced previously by Map functions (L2<Key2, Value2> instances) to populate them with the relevant list according to its indexation.

This process is considered as the principal phase in the current step (conversion operation), and it is realized via the Reduce functions specified in Algorithm 3, based on a set of conversion rules mentioned below which summarize the correspondence between each object Oi in data (L2) and the targeted CF-Tables:

- ✓ **Rule 1:** Each instance of List L2 will be converted to CF label of CF-Table ($Index_{L2}$ = $Label_{CF}$)
- ✓ **Rule 2:** For each instance of L2:
  - **Key2 =** CF-Table Columns.
  - **Value2**= CF-Table Columns Values.

Algorithm 3: Reducing algorithm
1: **Input**: list L2<Key2, list Value2>
2: **Output**: Column-Oriented table (CF-Table)
3: **Begin**
4: Index ← L2.getIndex()
5: CF-Table.setCFLabel(Index)
6: **Foreach** L2 **Do**
7: **If** (CF-Table.getColumns().isEmpty() = True) **Then**
8: CF-Table.addColumns(Key2)
9: CF-Table.addCorrespValues(Value2)
10: **Else**
11: DiffColumns ← SelectDiffCol (Key2, CF-Table.getColumn())
12: CF-Table.addCol(DiffColumns)
13: CF-Table.addValue(Value2).ON(Key2)
14: **End If**
15: **End Foreach**
16: **Return** CF-Table
17: **End Algorithm**

4.2 Transformation phase:

The transformation stage arrives after a successful extraction operation. It consists of handling the extracted data by applying several simple and complex operations such as Select, Join, Project and Union. These operations are classified as two different operation families, named 'Unary operation' that includes Select and Project operations. At the same time, the second type called the 'Complementary operation' that includes Union and Join operations. Both types of above operations will process a large amount of data. They will waste considerable time and effort with classical ETL logic, which motivated us to implement the MapReduce paradigm at this stage. The same served us remarkably minimizing the processing time, thanks to the parallelism aspects that support several processors to handle input data as divided tasks simultaneously, besides the scalability and fault tolerance offered by this paradigm.

In our approach, we adopt the transformation phase to use the Map and Reduce functions, which we applied primarily on a Select operation as it is the most known, used, and common query that allows us to select data as one or many columns from Column-oriented structures. The Select operation basic syntax is as follows: $\sigma(p)_{CF}$, where p represents the selection condition or predicate, and CF represents the different tuples we have in the column families. However, the p can also contain a restriction of data if we combine it with a Where clause, eventually the result of a Select statement will be a set of records from the provided CF.

The Project operation still also the same as a Select operation from a technical point of view, and it is represented as π(Col1, Col2), where Col1, and Col2 are the name of columns to be selected from the concerned column families.

Using ETL was never limited to execute Select and Project operations at the transformation stage. The requirement is more complicated than usual when it comes to aggregate data and apply different filters to produce more explicit and adaptable reports and insights. This reason has motivated us in this article to focus more on Join's operation due to its essential use on large datasets.

© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Algorithm 4: Transformation Phase
1: **Input**: List L3<CF-Table>
2: **Output**: J-Table (Joined table)
3: **Begin**
4: Q ← Parsing Query
5: list CF-Table ← getProjectedCF(L3)
6: list RC ← getRequestedColomns(Q)
7: list JoinConditions ← getJoinCond(Q)
8: **Call** MapProjection(CF-Tables,RC)
9: **Call** ReduceJoin(projected-Table1, projected-Table2, JoinConditions)
10: **Return** J-Tables
11: **End Algorithm**

The Algorithm below is responsible for performing the Map function, in which the system will take each Column family table (CF-Tables) as input, including the list of the requested columns (RC) that needs to be selected or projected. Then we will glance through each CF-Table searching for the RC to construct our final Key/Value List (L4) whenever a correspondence between the RC and the CF-Table is founded.

Algorithm 5: Mapping Function
1: **Input**: CF-Table, List RC
2: **Output**: List L4<Key4, List Value4>
5: **Begin**
6: **For** k ← 1 **to** RC.size() **Do**
7: **For** j ← 1 **to** M **Do** *//M represent the number of columns in the CF-table*
8: **If** (RC[k] = CF-Table[1,j]) **Then**
9: Key4 ← RC[k]
10: **For** i ← 1 **to** N **Do** *//N represents the number of lines in each CF-Table*
11: Value4.add(CF-Table[i,j])
12: **End For**
13: **Break**
14: **End If**
15: **End For**
16: **End For**
17: **Return** L4
18: **End Algorithm**

The second part refers to the Algorithm 6 of Reducing function, in which the system will take at least two Lists (L4-1 & L4-2) generated after the Map functions and browse all of them searching for the Joined condition in common above two Lists from the input join conditions parsed previously during the Algorithm 5. Then a Merge operation will be executed in order to Join the Input Lists (L4-1, L4-2) based on the Join condition.

Algorithm 6: Reducing Function
1: **Input**: List L4-1, L4-2, JoinConditions
2: **Output**: List L5<Key5, List Values5>
3: **Begin**
4: List L4-1-Keys ← L4-1.getKeys()
5: List L4-2-Keys ← L4-2.getKeys()
6: JoinCondition ← FilterCondition(JoinConditions).ON(L4-1.Keys, L4-2.Keys)
7: L5 ← Merge (L4-1, L4-2).ON(JoinCondition)
8: **Return** L5
9: **End Algorithm**

5. ILLUSTRATED EXAMPLE

In the below section, we are presenting an example that will help clearly understand our approach. We consider integrating Facebook Posts into our targeted data warehouse. Each Facebook Post is generated as a JSON file and subdivided into several sub JSON files (User.Json, Post.json, Location.json...etc) using the Splitting function presented previously in Algorithm 1.

Each Map function illustrated in Algorithm 2 will operate on a single sub JSON file, which is represented as <Key1, Value1> where Key1 is the name of the sub JSON file that corresponds to a dimension (Di) or fact (F), while Value1 represents the content of the sub Json file as presented in the Figure 4 (User.json file), and Figure 5 (Facebook Post.json).


```
{
  "id": "1",
  "age": 30,
  "first_name": "Abdeljalil",
  "last_name": "Boumlik",
  "gender": "male",
  "email": "boumlik.abdeljalil@gmail.com"
},
{
  "id": "2",
  "age": 28,
  "first_name": "Nassima",
  "last_name": "Soussi",
  "gender": "female",
  "email": "nassima.soussi@gmail.com"
},
{
  "id": "2",
  "first_name": "Mohamed",
  "last_name": "Bahaj",
  "gender": "male",
  "email": "mohamedbahaj@gmail.com",
  "location": "settat"
}
```


Fig.4 Description of User.json file

Journal of Theoretical and Applied Information Technology
15th September 2020. Vol.98. No 17
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
{
  "id_post": "1",
  "date_create": "20/06/2020",
  "user_create_id": "1",
  "link": "https://web.facebook.com/id_post=1id=1",
  "Post_text": "Our new paper related to ETL and Big Data",
  "Location": "Casablanca"
},
{
  "id_post": "2",
  "date_create": "21/06/2020",
  "user_create_id": "2",
  "link": "https://web.facebook.com/id_post=2id=2",
  "Post_text": "New post on Facebook about ETL and Big data",
  "Location": "Khouribga"
},
{
  "id_post": "2",
  "date_create": "22/06/2020",
  "user_create_id": "3",
  "link": "https://web.facebook.com/id_post=3?id=3",
  "Post_text": "New chalenge for ETL and Big Data",
  "Location": "Settat"
}
```


Fig.5 Description of Post.json file

Each Json file from the above example will be processed using Map function to construct L2-1 and L2-2 lists organized as <Key2, list Value2> presented in Table 2 and Table 3.

Table 2: User L2-1<Key2, list Value2>

Key2	List Value2
{id, age, first_name, last_name, gender, email}	{"1",30,"Abdeljalil", "Boumlik", "male", "boumlik.abdeljalil@gmail.com"}
	{"2",28,"Nassima", "Soussi", "female", "nassima.soussi@gmail.com"}
{id, first_name, last_name, gender, email, location}	{"3", "Mohamed", "Bahaj", "male", "mohamedbahaj@gmail.com", "settat"}

Table 3: Post L2-2<Key2, list Value2>

Key2	List Value2
{id_post, date_create, user_create_id, link, Post_text, Location }	{"1", "20/06/2020", "1", "https://web.facebook.com/id_post=1id=1", "Our new paper related to ETL and Big Data","Casablanca"}
	{"2","21/06/2020", "2", "https://web.facebook.com/id_post=2id=2", " New post on Facebook about ETL and Big data","Khouribga"}
	{"3","22/06/2020", "3", "https://web.facebook.com/id_post=3id=3", "New chalenge for ETL and Big Data","Settat"}

Once the Map functions complete the construction process of L2-1 and L2-2…etc., the Reduce function will be triggered automatically to convert these later into CF-Tables (User CF-Table, Post CF-Table) taking into account the multidimensional concept. Table 4 shows the example of User CF-Table.

Table 4: User CF-Table

User-Table			
Id	1	2	3
Age	30	28	
First_name	Abdeljalil	Nassima	Mohamed
Last_name	Boumlik	Soussi	Bahaj
Gender	Male	Female	Male
E-mail	boumlik.abdeljalil@gmail.com	nassima.soussi@gmail.com	mohamedbahaj@gmail.com
Location			Settat

In the current state, we finish the extraction phase by producing the final CF-Tables, then we continue with our second processing phase which is Transformation phase presented previously with Algorithm 4.

The first step consists to prepare the Joined tables with requested columns (RC) based on the Map Function presented in Algorithm 5.

In our case, we used the Select/Project operations treated on the Map functions in order to retrieve specific attributes related to any Facebook Post published by a given User Id. After that, the Reduce functions presented in Algorithm 6 will take as input the last generated tables (User CF-Table, Post CF-Table) in order to execute the Join operation and retrieve the requested results as per Table 5.

Table 5: Joined table

id	first_name	post_id	post_text	date_ create
1	Abdeljalil	1	Our new paper related to ETL and Big Data	20/06/ 2020

6. ANALYSIS AND DISCUSSION

This paper presents a new approach in the context of big data warehousing by applying advanced technologies into the ETL process that is performed sequence and adds the parallelism aspect in all his operations as Extraction and Transformation to support scalability and especially to minimize time-consumption. However, and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

based on the comparative studies and critical analysis carried out on the most recent approaches in Related works section, we have concluded that all existing approaches did not cover the partitioning and selecting data vertically according to a multidimensional structure, besides, they neglected the essential ETL operation.

In order to overcome the previous limitations, we have contributed with the current work in the improvement of ETL processing that needs serious attention to deal with the massive explosion of data to prepare and handle it into the DW. In fact, we readapt modeling ETL operations in the formal level of extraction and transformation phases to support the most used operations for treating and filtering data such as Select and Join with a specific focus on reducing side-join operation which was not treated before, while it's one of the most important functionalities used for processing large data sets. Our contribution is very helpful for producing relevant information for decision-makers and online Business Intelligent applications based on the MapReduce paradigm considered as a powerful solution for parallel processing of massive data.

7. CONCLUSION

In this paper, we have proposed a smart ETL approach for building a Data Warehouse from Big Data sources like Twitter and Facebook, also to conserve the multidimensional structure of Data Warehouse via the vertical partitioning of selected data from HBase as a column-oriented database. In fact, we have integrated the MapReduce paradigm in our ETL process to guarantee parallel processing during the extraction phase and also in the transformation phase by joining multiple tables in a reduce side due to its high importance in processing large data sets. Also, we manage to handle the most important features of ETL, which are Select, Project and Join operations, that are used frequently to operate on separated tables and to aggregate their data to generate analytic reports and insights.

As future work, and regarding the transformation phase of our smart ETL, we intend to support more operations to deal with complex processes such as Aggregation, Union, and different types of Join which will be considered as a step that may be taken further. Also, consider the performance factor that needs to be permanently enhanced. We aim also to adapt our solution to take into consideration new domains like the internet of things (IoT) which deals with data received from various devices, also in the medical domain that maintains huge data and requires multiple analytics.

REFRENCES:

[1] Mallek, H., Ghozzi, F., & Gargouri, F. (2020). Towards Extract-Transform-Load Operations in a Big Data context.

[2] Jo, J., & Lee, K.-W. (2019). MapReduce-Based D_ELT Framework to Address the Challenges of Geospatial Big Data.

[3] Saleem, Y., Crespi, N., Rehmani, M. H., & Copeland, R. (2019). Internet of Things-aided Smart Grid: Technologies, Architectures, Applications, Prototypes, and Future Research Directions.

[4] Zdravevski, E., Lameski, P., Dimitrievski, A., Grzegorowski, M., & Apanowicz, C. (2019). Cluster-size optimization within a cloud-based ETL framework for Big Data.

[5] R Baker, P MacHarrie, H Phung, J Hansford, J Reddy, S Causey, J Sobanski, S Walsh, R Niemann, D Beall -2019-Amazon Web Services (AWS) Cloud Platform for Satellite Data Processing

[6] Mallek, H., Ghozzi, F., Teste, O., & Gargouri, F. (2018). BigDimETL with NoSQL Database.

[7] SMF Ali - DOLAP, 2018. Next-generation ETL Framework to Address the Challenges Posed by Big Data.

[8] Boumlik, A., Soussi, N., & Bahaj, M. (2018). Automatic Data Modeling Transformation Approach Of Nosql Document And Column Stores To Rdf. Journal Of Theoretical & Applied Information Technology, 96(15).

[9] Mallek, H., Ghozzi, F., Teste, O., & Gargouri, F. (2017). BigDimETL: ETL for Multidimensional Big Data.

[10] Glushkova, D., Jovanovic, P., & Abelló, A. (2017). Mapreduce performance model for Hadoop 2.x. Information Systems

[11] Bala, M., Boussaid, O., & Alimazighi, Z. 2017). A Fine-Grained Distribution Approach for ETL Processes in Big Data Environments

[12] Boussahoua, M., Boussaid, O., & Bentayeb, F. (2017, August). Logical schema for data warehouse on column-oriented NoSQL databases. In International Conference on Database and Expert Systems Applications (pp. 247-256). Springer, Cham.

[12] Dehdouh, K. (2016, September). Building OLAP cubes from columnar NoSQL data warehouses. In International Conference on

© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Model and Data Engineering (pp. 166-179). Springer, Cham.

[14] Scabora, L. C., Brito, J. J., Ciferri, R. R., & Ciferri, C. D. D. A. (2016, April). Physical data warehouse design on NoSQL databases. In Proceedings of the 18th International Conference on Enterprise Information Systems (pp. 111-118). SCITEPRESS-Science and Technology Publications, Lda.

[15] Dehdouh, K., Bentayeb, F., Boussaid, O., & Kabachi, N. (2014, September). Towards an OLAP environment for column-oriented data warehouses. In International Conference on Data Warehousing and Knowledge Discovery (pp. 221-232). Springer, Cham.

[16] Bala, M., Boussaid, O., & Alimazighi, Z. (2014, November). P-ETL: Parallel-ETL based on the MapReduce paradigm. In 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA) (pp. 42-49). IEEE.

[17] Chung, W. C., Lin, H. P., Chen, S. C., Jiang, M. F., & Chung, Y. C. (2014). JackHare: a framework for SQL to NoSQL translation using MapReduce. Automated Software Engineering, 21(4), 489-508.

[18] Mohanty, S., Jagadeesh, M., & Srivatsa, H. (2013). Big data imperatives: Enterprise 'Big Data'warehouse, 'BI'implementations and analytics.

[19] Liu, X., Thomsen, C., & Pedersen, T. B. (2013). ETLMR: a highly scalable dimensional ETL framework based on mapreduce. In Transactions on Large-Scale Data-and Knowledge-Centered Systems VIII (pp. 1-31). Springer, Berlin, Heidelberg.

[20] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In 2011 6th international conference on pervasive computing and applications (pp. 363-366). IEEE.

[21] Dittrich, J., Quiané-Ruiz, J. A., Jindal, A., Kargin, Y., Setty, V., & Schad, J. (2010). Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). Proceedings of the VLDB Endowment, 3(1-2), 515-529.

[22] Blanas, S., Patel, J. M., Ercegovac, V., Rao, J., Shekita, E. J., & Tian, Y. (2010, June). A comparison of join algorithms for log processing in mapreduce. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (pp. 975-986). ACM.