Data Quality for Web Log Data Using a Hadoop Environment

QISHAN YANG, Insight Centre for Data Analytics, School of Computing, Dublin City University, Dublin, Ireland (qishan.yang@insight-centre.org)
MARKUS HELFERT, Insight Centre for Data Analytics, School of Computing, Dublin City University, Dublin, Ireland (markus.helfert@dcu.ie)

• Information systems → Data management system • Computing methodologies → Distributed computing methodologies

Solving data quality problems is important for data warehouse construction and operation. This paper is based on developing a web log warehouse. It proposes a data quality problem methodology for data preprocessing within the log warehouse. It provides a hierarchical data warehouse architecture that is suitable for resource saving and ad hoc requirements. The data preprocessing is completed using Hadoop associated with its sub-projects such as Hive, HBase etc. In this paper we compare a Hadoop setup with a Oracle based architecture.

Additional Key Words and Phrases: Data Quality, Data Preprocessing, Data Warehouse, Web Log, Hadoop, Hive, HBase, Oracle

1. INTRODUCTION

As institutions' data centers, data warehouses store a vast number of historical data to fulfill data mining and analysis requirements. According to [Inmon et al. 2010], a data warehouse is a subject oriented, integrated, non-volatile and time-variant management decision supporting data repository. Raw data might be gathered, cleansed and integrated from heterogeneous data resources before be populated into data warehouses. This operation of data preparations cannot be ignored in order to ensure the data quality.

Data quality problems usually come first when data need to be analyzed. It occupies approximately 80% of the total data engineering effort in practice [Zhang et al. 2003]. For example, a large amount of time is spent on how to preprocess unformatted or defective data. After this, the processed data would be analyzed, mined or utilized to make decisions as input sources. The data preprocessing might be the last inspection to handle data quality problems before data is delivered into data warehouses.

The data from RDBMS normally has referential integrity constraints which enforce the business rules associated with databases and prevent the entry of invalid information into tables [Oracle Help Center 2016]. It also has pre-defined data models. Hence, structured data could be easier quality assured. However, unstructured data is approximately five times more than structured data [Inmon 2006]. The structured data is not sufficient for decision-makers or data analysts, if they want to get more benefits from data. Web log data is textual data generated by web servers, which is classified into the unstructured data [Zicari 2014]. As the website archive, web log files store a vast number of interactions between website servers and clients. They also include massive outliers, noises and dirty records that need to be scrubbed.

The aim of this paper is investigating data quality problems and preparing data for a web log data warehouse. The contribution is proposing a data quality problem methodology for data preprocessing within the log warehouse and providing a hierarchical data warehouse architecture that is suitable for resource saving and ad hoc requirements. In addition, it makes a comparison between the data warehousing establishment in the Hadoop environment and Oracle. It does not refer to web log

mining operations such as how to identify users, user sessions, path completions, patterns etc. The structure of this paper is organized as follows. Section 2 provides the related research in this field. Section 3 gives some background of the technologies used in this research. Section 4 provides the experimental setup. The detail of the data cleansing and classification is described in section 5. Section 6 illustrates the hierarchical data warehouse architecture. Section 7 offers the evaluation of this approach. The conclusion is demonstrated in section 8.

2. RELATED RESEARCH

Some research has been done in the field of the web log data preprocessing. For example, [Castellano et al. 2007] implemented the LODAP (Log Data Preprocessor) in Microsoft ACCESS database. This tool had three modules: the data cleaning, the data structuration and the data filtering, in order to remove useless records, identify user sessions and the most visited pages respectively. [Kherwa and Nigam 2015] provided a milestone of the data preprocessing for the web usage mining. According to their paper, the previous research in this field more focused on specific purposes such as user and session identifications. However, our paper is mainly to prepare data for the web log data warehouse which can fulfill the most of the business requirements.

There is some work has been done in the field of the web log data warehouse implementation. [Özcan et al. 2011] used Hadoop to prepare data then delivered the processed data into BD2 which is a database server. [Thusoo et al. 2010] built a data warehouse architecture by using Scribe, Hadoop and Hive at Facebook. In contrast to previous works, we focus on the data preparation and a novel data warehouse structure using Hadoop, HBase, Hive etc. due to data quality and resource saving considerations.

In terms of the partitioned data warehouse, [Bellatreche et al. 2000] split the dimension tables then separated the fact table into several fragments based on the dimensional fragmentation schemas. There were some fragmentation selection algorithms mentioned by [Thenmozhi and Vivekanandan 2014] to form the fragments, which include the Hill Climbing, Genetic Algorithm, GAHC (the combination of the Hill Climbing and Genetic Algorithm algorithms) and GATS (Genetic Algorithm with Tabu Search). If business requirements are clearly identified before establishment of the data warehouse, we may not need to populate all data into the data warehouse and split them. It may save time and avoid data migrations.

3. BACKGROUND

3.1 Data Warehouse Architecture

In terms of the data warehouse architecture, the traditional and well referenced three-layered data architecture is used as an underpinning concept in our research. The architecture comprises real-time data, reconciled data and derived data [Devlin and Cote 1996]. It is depicted as following in Fig 1. In an initial phase, this paper only focuses on the real-time layer and reconciled layer.

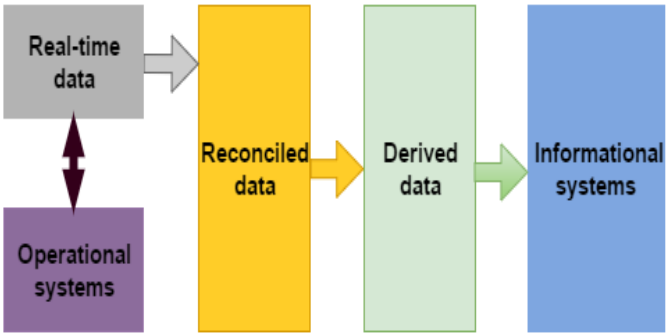

Fig. 1. The three-layered data warehouse architecture [Devlin and Cote 1996]

3.2 Oracle

According to [Ariyachandra and Watson 2006], the most popular and frequently used platform for data warehouses was Oracle which occupies 41 proportions in this field, followed by Microsoft (19%) and IBM (18%).

3.3 Hadoop

Hadoop is a platform which has the ability to parallel process the distributed large data in independent computing nodes by using simple programming models. It is a distributed and highly analytical tool associated with HDFS (Hadoop Distributed File System) and MapReduce to place and manipulate data. HDFS has the high-throughput capability to store and retrieve data. Yarn is a job scheduling system to manage cluster resources in the Hadoop environment. MapReduce is a system to parallel process large data sets organized by Yarn [Hadoop 2016]. A number of the international companies such as Apple, eBay, LinkedIn, Yahoo and Facebook have been using this framework or similar platforms [Holmes 2012].

3.4 Hive

Hive provides the data warehousing functions to query and manage enterprise-sized datasets in distributed file systems. It has a mechanism to operate the data warehouse by using a SQL-like language called HiveQL rather than users write MapReduce programs. Meanwhile, it also offers map/reduce to extend ad hoc functions by plugging in customized mappers and reducers if the HiveQL cannot fulfill the requirements [Hive 2016].

3.5 HBase

HBase is a NoSQL database system to organize large datasets and can manage very large tables with billions of rows millions of columns. It is an open-source, distributed, non-relational database based on the Google's Bigtable. This database can be seamlessly set up in Hadoop environments and fully leverage the mechanisms of MapReduce, HDFS, Yarn etc. [HBase 2016].

3.6 Flume

Flume provides a distributed service for efficiently extracting and delivering large amounts of log data from sources to destinations. It is a highly-available and fault-tolerant tool associated with mechanisms such as failover and recovery. This extraction software also has some features to collaboratively work with other tools in Hadoop environments [Flume 2016].

4. EXPERIMENTAL SETUP

In our experiment, data sets have been divided into different classifications before populating them into the multi-level data warehouse. It did not depend on the dimension tables to split fact tables. The dimension tables in this research were not split, because the small size of data has been stored in them. Besides, this data warehousing architecture not only contains fact tables and dimension tables but also is composed of plain files as the standby data sets for special requests. If ad hoc requirements need to be fulfilled, these data resources are populated into their related fact tables.

The aim of this experiment was preprocessing data and evaluating the data warehousing establishment and loading in a Hadoop environment and Oracle. The Hadoop environment was built on a physical machine (Intel Core i7 CPU 3.60GHz) associated with nine Linux virtual machines. Three of them were set up as data resources. The rest of them were used to build the Hadoop platform. Its configuration information is presented in table I.

Table I. The Hadoop Platform Configuration

Node Name	Operation System	Processor	Memory	Software
Node1	Centos 7 X86_64	1	1GB	JDK 1.7, Hadoop, HBase, Hive
Node2	Centos 7 X86_64	1	1GB	JDK 1.7, Hadoop, HBase
Node3	Centos 7 X86_64	1	1GB	JDK 1.7, Hadoop, HBase
Node4	Centos 7 X86_64	1	2GB	JDK 1.7, Hadoop, HBase, Zookeeper
Node5	Centos 7 X86_64	1	2GB	JDK 1.7, Hadoop, HBase, Zookeeper
Node6	Centos 7 X86_64	1	2GB	JDK 1.7, Hadoop, HBase, Zookeeper

The Hadoop (version 2.7.1) was deployed based on HDFS High Availability. The Yarn was running in Node1. The tow NameNodes were configured in Node2 and Node3. The testing data were placed in DateNodes (Node4, Node5 and Node6). The comparative trial was built on Oracle Database 12c in the same physical machine. The programming language used in both of platforms was Java based on JDK 1.7.

5. DATA CLEANSING AND CLASSIFICATION

We cleansed and classified the original data in the Hadoop environment in this experiment. The main reasons are given as follows. The parallel computation of Hadoop could be utilized to enhance the speed of the operations. The result would be stored in HDFS, which would be manipulated by MapReduce or Hive directly. The outline and the whole data flow from data sources to manipulation areas are presented in Fig. 2. which also includes the data flow of the data warehousing construction and loading.

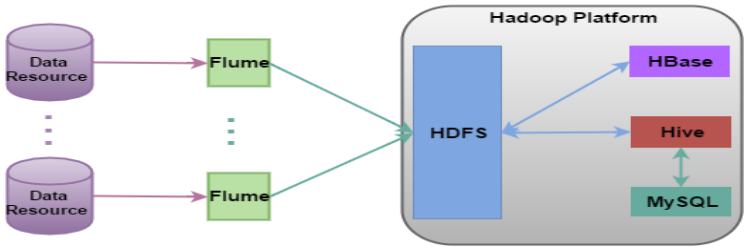

Fig. 2. The outline and data flow

5.1 Data Extraction

Data extraction is a part of the ETL processing. In this system, Flume was set up to fetch and deliver web log files from severs to HDFS. It did not cleanse and format data. It collected log data from physically independent servers into HDFS. The procedure is described as following. The Linux shell script was run every midnight automatically to create a folder in HDFS, which was named the date of yesterday. Flume extracted yesterday's log files then sent them into this folder.

5.2 Mapper

After the data had already put into the folder, the MapReduce was invoked to process log documents. In the map function, the log records were manipulated and classified. Figure 3 illustrates the detailed program flow diagram in the map function.

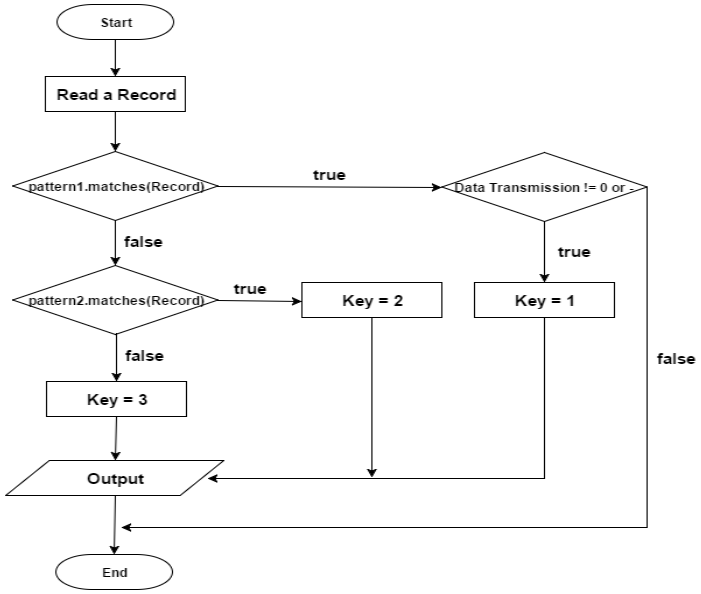

Fig. 3. The program flow in the map function

At the start of the experiment, a certain number of Mappers were initialized based on how many the blocks did the log files have. Then these Mappers would read records in related blocks. After this, each record would be examined by two regular expressions. The first regular expression was to check invalid interactions (e.g. status > 299) between clients and servers, if it returned true then checking the data transmission, if their data was not equal to "0" or "-" then setting these records' key to 1 and output them otherwise ending this processing. The job of the second one was to identify the effective records such as querying assistant resources (js files, css files, pictures etc.), and if returned true, then assigned 2 to their key and output them. If all regular expressions returned false, these records were turned out to be valuable entries, then set their keys to 3 and output them.

5.3 Reducer

After the Mappers, each record had a key in the scope of 1, 2 or 3. Reducers did not need to handle the complicated workflow as Mappers did. The main task of them was distinguishing key of each record and classifying them. After all reducers finished

their jobs, the classified records had been loaded into three different files. The program flow in the reduce function is presented in Fig 3 as following.

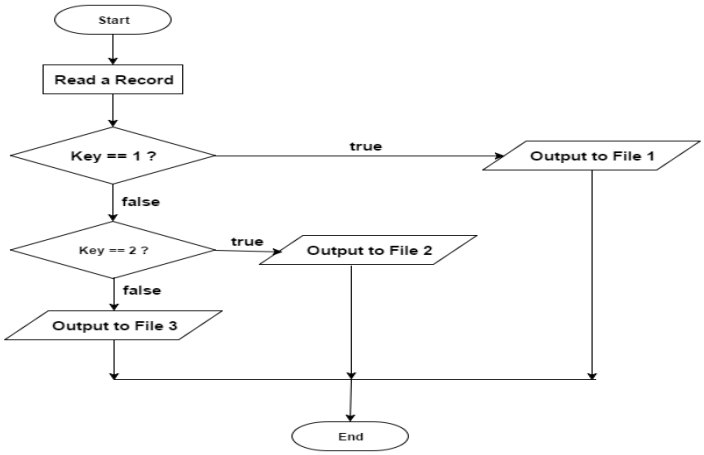

Fig. 4. The program flow in the reduce function

6. HIERARCHICAL DATA WAREHOUSE

Fact tables depend on foreign keys to connect their dimension tables. Hence, when inserting business facts into them, it is necessary to visit dimension tables in order to acquire foreign keys. Sometime, if dimension tables did not have the related records, new entries would be inserted into corresponding dimension tables. It is time-consuming to load data into fact tables, especially, if the data sets are unstructured.

The data in fact tables overwhelmingly occupies the largest part of volumes in any data mart [Kimball & Ross 2011]. If the raw data is cleansed, classified and selectively populated into related fact tables, maybe it is an effective choice to reduce the volume of fact tables. The whole processing is described as follows: in the beginning, classifying the raw datasets into different weighted levels based on the business requirements, after this step, preparing and populating the most valuable data set into the data warehouse then second important data etc. Figure 5 presents the data flow of this data warehouse architecture.

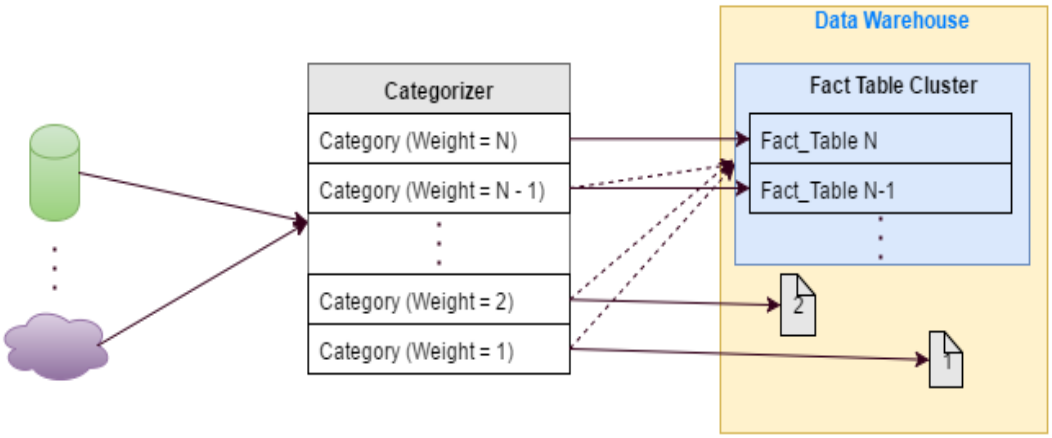

Fig. 5. The hierarchical warehouse data flow

If the ad hoc queries are in the to-do-list, the related datasets would either be loaded into the fact table or the extended fact table, even be manipulated using other approaches (e.g. MapReduce) to archive the goals without loading the data sets into the fact tables. This data warehouse architecture would reduce resources expenditures and keep fit for the sizes of fact tables. It may also suit for other unstructured data sets, because a vast volume of unstructured data may contain a great many outliers or unimportant information in some cases, while, sometime, these irrelevant records are still useful. For example, if the total data flow of a website need to be evaluated, all records even error entries need to be taken into account. The hierarchical data warehouse structure can fulfill most of the business requirements even ad hoc queries. The example of the architecture is demonstrated in figure 6 below.

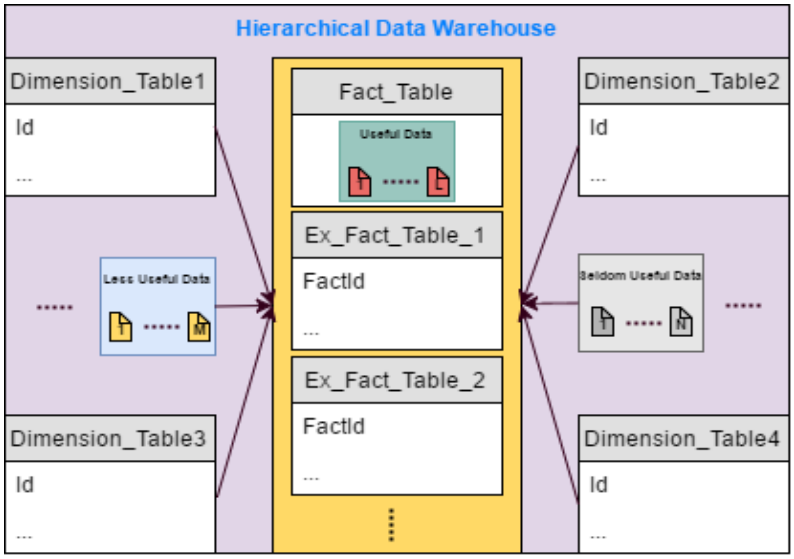

Fig. 6. The hierarchical data warehouse architecture

7. EVALUATION

7.1 Experiment

The volume of the data used in this experiment was 1 GB which contained 11882090 records. For the sake of saving resources and fulfilling the ad hoc requirements, almost all records were classified into three categories by Mappers and Reducers: the frequently used data (weight = 3), the less frequently used data (weight = 2) and the seldom used data (weight = 1). Table II lists the detailed information of datasets and the performances of data manipulations in the Hadoop environment and Oracle Database.

Table II. Data sets and performances in two platforms

Data Set	Size (MB)	Data Query (MB)	Hadoop (Mins)	Oracle (Mins)
Frequently Used	86.20	8,139.39	97.27	54.59
Less Frequently Used	816.00	75,957.32	Do not Need to Load	20.12
Seldom Used	121.80	24.81	Do not Need to Load	3.71
Total	1024.00	84,121.52	97.27	78.42

The fourth column showed the data formatting using MapReduce and HBase in the Hadoop platform. The main jobs of HBase in the Hadoop environment were helping MapReduce to generate LogId, PathId, DomainId etc. and forming dimension tables. If new record came, it would be checked whether related information had already stored in the dimension tables or not. In the Oracle column, it provided three-leveled datasets formatting and populating into Oracle Database. As can be seen from this table, the frequently used data set size was only 86.2 MB out of 1024 MB (around 8.4180%). Less frequently used data set occupies the most volume of the raw log files 816.00 MB (79.6875%), followed by the seldom used data 121.80 MB (approximately 11.8945%).

In the Hadoop environment, it took 97.27 minutes to format the frequently used dataset and generate surrogate keys for LogId, PathId, DomainId etc. by checking HBase, then output them into files stored in HDFS. However, in Oracle database, it spent 54.59 minutes to do the same things and insert fact records into the fact table. The Hadoop did not need to spend time to insert the fact table into its related fact table, but it needed to form fact records plain files and put them in a folder which the fact table (on Hive) had already pointed to it as a data source directory. The web log data warehousing schema and its set-up were similar to the schema in [Yang and Helfert 2016] which was built in a Hadoop environment, while the schema in this experiment had multi-level fact tables.

In Oracle platform, it took 20.12 and 3.71 minutes to load less frequently used and seldom used datasets respectively. These operations were much fast than the frequently used set loading. The main reason was these operations did not need to visit dimension tables or create surrogate keys. It only needed to format and insert records into extended fact tables. Whereas in the context of Hadoop, these operations were not necessary, only just delivering these related files into the folders which belonged to the tables in Hive.

The operation in the Hadoop environment did not need to load all levels' datasets. It only strictly formatted frequently used records and wrote them into files then pointed out the rest of the levels' files in HDFS for extended fact tables. However, the warehouse in Oracle needed to load all data into the related fact tables in order to respond to the ad hoc queries. The disadvantage of the Hive based data warehouse in Hadoop was that it needed extra helpers (MapReduce) to analyze data such as counting the total data flow of the website. However, the warehouse in Oracle did not need any assistant to handle this kind of ad hoc queries.

7.2 Generality of the Approach

In order to find out the threshold and generalize this approach, the frequently used data has been divided into several groups. To be more specific, it has been split into 5 groups by using part of the frequently used data which occupied 1%, 3%, 5%, 7% and 8.4% of the original data set. Rest of the data was acted as the infrequently data. For example, if the frequently used data had 1%, then the infrequently used data occupied 99%. Then these partitioned data sets would be manipulated even loaded into related tables (e.g. loading data into the fact tables in Oracle). The diagram 7 illustrates the time taken including processing the frequently and infrequently used data in each group.

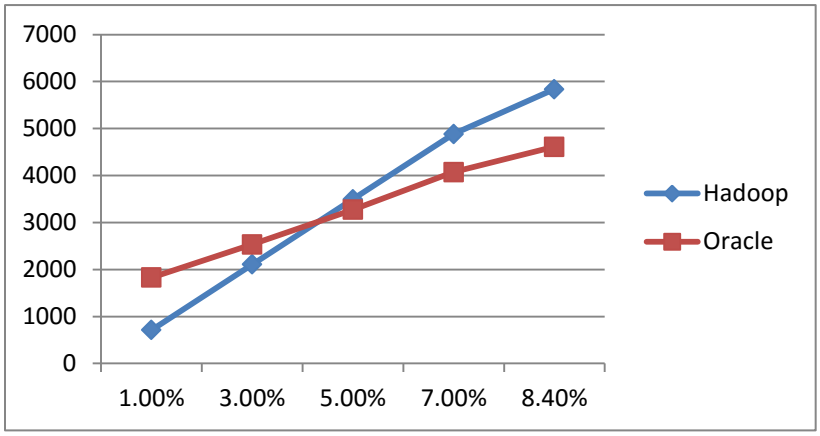

Fig. 7. The performances in Hadoop and Oracle platforms with different data sets

As can be seen, when the frequently used data set was less than around 5%, the Hadoop environment was faster than the environment of Oracle in this circumstance. The main reason was that the frequently data needed to be formatted and infrequently used data needed to be loaded in Oracle. However, the Hadoop environment did not need to do this job, only formatting data because of the character of Hive. However, when the frequently used data set increased above 5%, Oracle spent less time to load data. Because the Hadoop environment required more time to manipulate frequently data set than Oracle needed.

In this experiment, it can be seen that when the frequently used data set was greater than 5%, Oracle was faster than the Hadoop environment. However, it might be not proper to say Oracle had higher performance than the Hadoop environment had in all situations. We only tested the performances in this particular case and set-up. There were other capabilities may also need to be considered. For example, it did not test the flexibility. If the hardware needs to be upgraded, the Hadoop environment is much easier to make this happen compared with the Oracle.

8. CONCLUSIONS

Data quality problem solving is the indispensable task for the data warehouse building and the data mining. It is time-consuming to preprocess data and guarantee data quality. This paper brings forward a methodology to prepare data for the data warehouse and a hierarchical data warehouse architecture to enhance the data manipulation and fulfill ad hoc requirements. This architecture could be extended to other unstructured data preparations and warehousing constructions.

There were still some eliminations existing in the experiment and novel warehousing architecture. The raw data classification policy should be known by analyzing business requirements before the manipulation of data. The raw data set was not sufficient for further testing the capabilities of this system. In addition, the Hadoop environment was set up on virtual machines which shared the sources of the host computer. In the following work, the Hadoop environment would be moved to physically independent computers. At the same time, more raw data would be used to test the throughput, flexibility and performance of this architecture in the context of big data.

ACKNOWLEDGMENTS

This publication was supported by Science Foundation Ireland grant SFI/12/RC/2289 to Insight- Centre for Data Analytics (www.insight-centre.org).

REFERENCES

Inmon, W.H., Strauss, D. and Neushloss, G., 2010. DW 2.0: The architecture for the next generation of data warehousing: The architecture for the next generation of data warehousing. Morgan Kaufmann.

Zhang, S., Zhang, C. and Yang, Q., 2003. Data preparation for data mining. Applied Artificial Intelligence, 17(5-6), pp.375-381.

Oracle Help Center. 2016. Database Concepts: 21 Data Integrity. (2016). Retrieved April 9, 2016 from https://docs.oracle.com/cd/B19306_01/server.102/b14220/data_int.htm

Inmon, B., 2006. DW 2.0; Architecture for the Next Generation of Data Warehousing. Information Management, 16(4), p.8.

Zicari, R.V., 2014. Big data: Challenges and opportunities. Big data computing, pp.103-128.

Devlin, B. and Cote, L.D., 1996. Data warehouse: from architecture to implementation. Addison-Wesley Longman Publishing Co., Inc..

Hadoop. 2016. Welcome to Apache Hadoop. (2016). Retrieved April 9, 2016 from https://hadoop.apache.org/

Holmes, A., 2012. Hadoop in practice. Manning Publications Co..

Hive. 2016. Apache Hive TM. (2016). Retrieved April 9, 2016 from https://hive.apache.org/

HBase. (2016). Welcome to Apache HBase. (2016). Retrieved April 9, 2016 from https://hbase.apache.org/

Flume. (2016). Welcome to Apache Flume. (2016). Retrieved April 9, 2016 from https://flume.apache.org/

Castellano, G., Fanelli, A.M. and Torsello, M.A., 2007. Log data preparation for mining web usage patterns. In IADIS International Conference Applied Computing (No. 10000, p. 20000).

Kherwa, P. and Nigam, J., 2015. Data Preprocessing: A Milestone Of Web Usage Mining. International Journal Of Engineering Science And Innovative Technology (Ijesit) Volume, 4.

Özcan, F., Hoa, D., Beyer, K.S., Balmin, A., Liu, C.J. and Li, Y., 2011, June. Emerging trends in the enterprise data analytics: connecting Hadoop and DB2 warehouse. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 1161-1164). ACM.

Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sen Sarma, J., Murthy, R. and Liu, H., 2010, June. Data warehousing and analytics infrastructure at facebook. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (pp. 1013-1020). ACM.

Bellatreche, L., Karlapalem, K., Mohania, M. and Schneider, M., 2000. What can partitioning do for your data warehouses and data marts?. In Database Engineering and Applications Symposium, 2000 International (pp. 437-445). IEEE.

Thenmozhi, M. and Vivekanandan, K., 2014, August. A comparative analysis of fragmentation selection algorithms for data warehouse partitioning. InAdvances in Engineering and Technology Research (ICAETR), 2014 International Conference on (pp. 1-5). IEEE.

Ariyachandra, T. and Watson, H.J., 2006. Which data warehouse architecture is most successful?. Business Intelligence Journal, 11(1), p.4.

Kimball, R. and Ross, M., 2011. The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons.

Yang, Q. and Helfert, M., 2016. Revisiting arguments for a three layered data warehousing architecture in the context of the Hadoop platform. In: The 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), 23-25 Apr 2016, Roma, Italy. ISBN 978-989-758-182-3