Cassandra and SQL Database Comparison for Near Real-Time Twitter Data Warehouse

Muh. Rafif Murazza
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia
muh.rafif.m@mail.ugm.ac.id

Arif Nurwidyantoro
Department of Computer Science and Electronics
University Gadjah Mada
Yogyakarta, Indonesia
arifn@ugm.ac.id

***Abstract*—In the era of Big Data, social media analysis has grown extremely popular. Twitter, one of the most popular social media, is believed to contain many user opinion in its message. Thus, leading organizations start utilizing its data using social media analytic tools to get in-sight of their markets in real-time. However, many Twitter analytic tools are still specified only in some specific tasks. Therefore, in order to enhance the possibility of doing many analysis on Twitter, a data warehouse technology can be utilized to receive, process, and store a real-time Twitter streams. Nonetheless, data warehouse development using relational database start to show its limit on storing big data let alone real-time. Hence, NoSQL (Not Only SQL) technology has emerged as an alternative solution. In this paper, we try to develop near real-time Twitter data warehouse using NoSQL database, Cassandra, and compare its storing and querying performance with that developed using relational databases. The results show that Cassandra performs significantly better in storing data than the relational databases. Meanwhile, in its querying performance, Cassandra is slower while using small data but way faster on vast data.**

Keywords— casandra; NoSQL database; social media; real time

I. Introduction and Motivation

In the recent years, the increase of internet users and the activities of social networking have caused the rapid growth of data which later known as Big Data [1] [2] [3]. Organizations see this phenomenon as an opportunity to gain insight of their markets since the social media data itself contains a lot of information of its users. Not only that, researcher have also begun to obtain the useful information of social media data for their research needs [4]. Thus, emerged a new area of data analysis, known as Social Media Analysis [2].

Social media analysis has become part of decision making process in many organizations or companies by using advance analysis technique to discover the patterns in the data [5]. Yet, it is believed that the existing social media analytic tools are specified in some specific tasks only, because exploring the whole activities from a large database is difficult and expensive [2] [6]. In order to provide an universal social media analytic tool, data warehouse technology with few preprocessing steps are required to store and process the unstructured data of social media [7] [8] [2].

However, data warehouses which usually implemented using RDBMS database are starting to show their limitations when it comes to storing and managing big data, let alone real-time [9]. Therefore, NoSQL databases which developed to support application involving big data have become popular [10] [11]. Eventually, NoSQL databases are starting to be considered as alternative to relational databases since it offers performance way higher than that of relational databases [12]. We believed that NoSQL databases could be the solution in develop-ing real-time data warehouse for social media.

This work mainly focused on developing near real-time Twitter data warehouse in NoSQL database, Cas-sandra, and comparing its performance with that devel-oped in relational databases. The data warehouse will receive, process, and store the Twitter data directly from live streams into the database. Each data warehouse will be implemented using a set of schema based on the characteristic of database and Twitter to observe the performance within some circumstances. We compare the write and read operations performance to see which of the selected databases are more suitable in developing real-time Twitter data warehouse.

In this paper, the rest of the introduction section gives an overview of the main components of this work. Section II reviews related works about NoSQL usage for data warehouse, Twitter data warehouse, and real-time data warehouses. Section III describes the underlying parameters, variables, and scenarios used in experiments.

978-1-5090-1709-6/16/$31.00 ©2016 IEEE

Section IV presents the experiments setup and results. The final section will summarizes the conclusion.

A. Twitter

Twitter is one of social media which developed to support fast communication [13]. Its simplicity and quick access in exchanging message has made Twitter as one of main communication in people's life. In addition to that, its users has reached 140 million with at least 400 million tweet per day [13]. As the increase of popularity, Twitter has attracted research from various background such as politics, business, academics, social, et cetera. Derived from that, Twitter has developed public APIs so that researcher could access their public data at will. Twitter API divided into REST API and Streaming API. REST API is used to collect information as requested by the user. Meanwhile, streaming API could provide new data continuously even when no requested are made.

B. Apache Cassandra

Apache Cassandra is one of NoSQL database based on project that developed a second generation of highly scalable and distributed database based on column [14]. The distribution system in Cassandra is a set of database nodes. Therefore, a write operation in Cassandra will be duplicated in another node, and a read request will be directed to certain nodes [15]. Meanwhile, among other things, the main characteristic of Cassandra and its selling points are its scalable and fault-tolerant ar-chitecture, versatile and flexible data model, declarative and user friendly query language (CQL), very efficient read and write paths, scale to millions of transactions per second, and easy failure handling [15] [16]. In terms of performance, Cassandra is claimed to be better than any other column-oriented NoSQL databases [16].

II. Related Works

Some business nowadays have needs of more dynamic information that traditional data warehouse is claimed to not be able to provide it which triggers the development of a real-time data warehouse [17]. Chen et al. propose a real-time data warehouse approach by focusing on its refresh mechanism [17] which is called the *incremental continuous refresh*. On the other hand, Zhu et al. [18] and Wang and Liu [19] approach a real-time data warehouse using Service-Oriented Architecture (SOA).

Rehman et al. [2] develop a data warehouse for Twitter while not focused on real-time approach but focused on Twitter data preparation and enrichment. In their work, they combine both NoSQL and SQL databases due to the massive data stream. The NoSQL database, BaseX, used as data buffer to convert, enrich, and filter the Twitter data before they are stored in the main data warehouse in SQL database. Even though the data warehouse is not yet real-time, they are able to receive and store the entire stream data.

Despite the high performance, NoSQL databases are still incapable when it comes to aggregate and join operations which is the reason why people still con-sider developing data warehouse in relational database. As a consequence, Dehdouh et al. provide a solution for aggregate operation on data warehouse developed in column-oriented NoSQL, HBase, namely CN-CUBE [9]. On the other hand, Carniel et al. [20] presents Star Schema Benchmark, a query solution for NoSQL data warehouse, and compare its query performance when implemented in FastBit and that in traditional data warehouse. Yet, those two works are not focused toward a real-time approach despite using NoSQL for data warehousing.

Every NoSQL database are unique individually. We need to understand its characteristic thoroughly before starting to design or even implement a data warehouse on it. Chebotko et al. [16] present a guide for data modeling in Cassandra. According to them, the concept of data modeling in Cassandra is very contrast from that in relational database. In summary, queries need to be considered in designing Cassandra schema design which relies on data nesting and denormalization. In addition that, unlike in relational database, it is normal to duplicate data in different Cassandra tables in order to support many queries.

This paper will contribute an observation of Cassandra performance compared to the two other popular relational database, MySQL and PostgreSQL. This work is mainly based on that of Rehman et al. [2] but excluding the data buffer and use NoSQL database instead of SQL in order to provide a near real-time data warehouse approach.

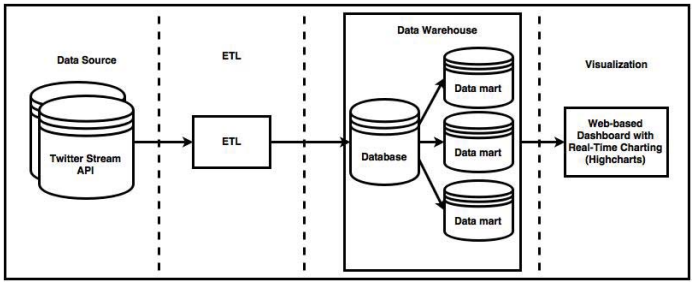

Fig. 1. Twitter data warehouse architecture

III. Data Warehouse Architecture and Experiment Setup

This section discusses the underlying architecture of the data warehouse and the parameters of the experi-ments. The architecture of this data warehouse presented in fig 1 is basically common, except for the data source. Instead of an OLTP database, the data comes from Twitter Stream API. Thus, in order to update a new data, the data warehouse is not required to refresh manually and only needs to process the new data from the streams directly and store it in the data warehouse. As the data is written in the database, it is also read in about the same time from the web-based dashboard to present it in a real-time charts. The visualization plays the role as the proof that the data warehouse could retrieve, process, and store Twitter data in near real-time.

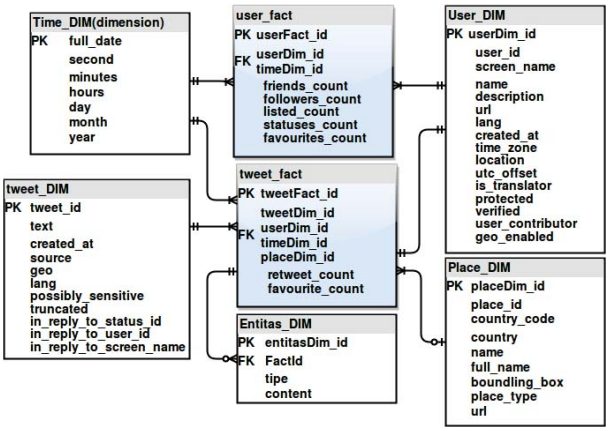

Fig. 2. SQL data warehouse schema

As the characteristics of Cassandra and the other two SQL database are in contrast, the data warehouse schema designs differ as well. Ideally, data warehouse developed in Cassandra should use a schema-less or denormalized main flat-table and several additional flat-tables each supporting predefined queries since it does not support join and aggregate operations [16]. On the other hand, data warehouse developed in SQL database should use star schema to reduce data duplication and improve the speed performance.

Fig 2 shows the star schema used in SQL database which should also be representing the Twitter entities cardinal relationship. The star schema used for the SQL database are based on the work of Rehman et al. [21] with some modifications. The modifications lay on the number of dimensions and fields as this research mainly focused on storing every data rather than only storing data essential for analysis purpose.

As have shown in Fig. 2, the cardinal relationship of each dimension table differs, specifically *Twitter external entities*. Since each tweet could consist several external entities, the *TweetFact id* is referenced in the dimension table instead of the other way like others dimension table.

On the other hand, data warehouse developed in Cassandra use *Collections* data type to cope with that Twitter characteristic. The *Collections* data type is not only used for *External entities dimension*, but also for *place dimension* and *boolean-typed fields*. The reason is to minimize the *null* fields from *Place dimension* as some Twitter users do not enable their *Geolocation* causing them to be *null*. Above that, it will use flat-table schema presented in Fig. 3 with several small flat-tables each supporting some queries.

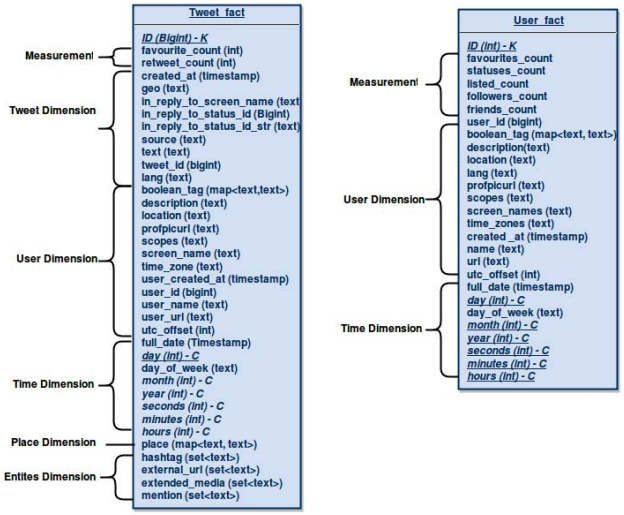

Fig. 3. Cassandra data warehouse schema

The experiments are divided into two parts, experi-ment on write operation and read operation. The write operation is the key to the real-time data warehouse, and in order to get a broad view of its performance comparisons, the experiment on write operation involves several conditions. In the contrary, the experiment of read operation do not involve any additional conditions. Both of the experiments will measure and record its operation time which later will be tested statistically.

There are two conditions for the experiments on write operation. The first experiment is to compare the data warehouse performance with the ideal design for all of them so that they are capable to manage the whole Twitter data. In the second experiment, both Cassandra and SQL databases will be implemented using flat-table for each fact (e.g tweet and user). Nevertheless, the Twitter external entities (e.g. hashtag, mentions, url,

media) is not involved in this experiment, because of its many-to-one relationship with the tweet which will cause massive data duplication in the SQL flat-table. In addition to that, saving data into a single table in SQL is believed to improve its write operation speed. The idea is to compare Cassandra with SQL while SQL has the condition for best speed possibility even with the absence of comprehensive value of the data.

During the experiments, the Twitter data streams have to be the same for each experiment conditions since they play the role as control variable. Thus, sample of Twitter data streams need to be collected beforehand so that it could be used and assumed as the real Twitter data stream in the experiments. The stream sample will be stored using MongoDB, a NoSQL database that capable in storing a whole JSON document without altering it since the Twitter data streams produces its data in JSON format.

Furthermore, each write operation experiment condi-tions are executed with several set numbers of tweet as another control variable and will be repeated ten times for each set of tweet. The numbers of tweet that will be processed are started from 10, 20, until 100 and 100, 200, until 500 which in total there are 14 set numbers of tweet. In a different manner, the read performance experiment is executed with only using 3 set numbers of tweet to be queried which are 1,000, 5,000, and 10,000 tweets.

Then, the experiments data will be compared and tested using Mood Median Test, because this type of experiments tend to have extreme outliers [22]. Mood Median Test is used to see whether there are any significant differences regarding the performance speed in all databases [22]. Beside the mentioned conditions for write operation, Cassandra from the second condition will also be compared with SQL database from the first condition in order to observe the write performance comparison even when the condition is in the advantage of the SQL database.

TABLE I.
SYSTEM SPECIFICATION

Name	Specification
Operating System	Ubuntu 14.04 LTS 64-bit
Temporary storage	MongoDB 3.0.8
Cassandra	Version 2.0.17
SQL Database I	MySQL Ver. 14.14 Distribution 5.5.46
SQL Database II	PostgreSQL Version 9.3.10
Processor	Intel CoreTM i5-5200U CPU @2.20GHz x 4
Memory	4GB

Table I describes the system specification used in the experiment. We do not use the latest version of Cassandra because its PHP driver is not yet compatible with the latest version. Additionally, Cassandra PHP driver is required in the web-based dashboard for the visualization. All of the databases use the default con-figuration and experimented in a single computer.

IV. RESULTS AND EVALUATION

The results of the first experiment are summarized in figure 4. Each number of result presented in the chart are the median from the 10 experiment data repetition. The first experiment measure the time taken in saving Twitter data while each of the databases implementing their ideal design to cover whole Twitter data.

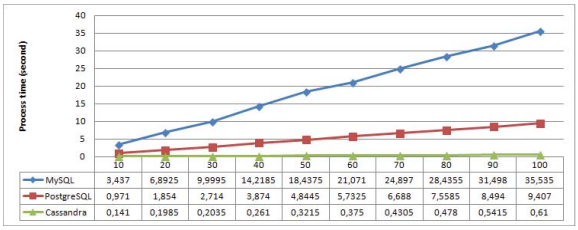

Fig. 4. Time for saving 10 - 100 Twitter data with the first condition

The results of the first experiment show that Cassandra offers the fastest write operation while MySQL is the slowest. In addition to that, the Mood Median test results show that the time differences made from each databases are statistically significant.

The second write experiment measure the time spent in saving Twitter data while all of the databases are implemented using flat-table and without involving Twitter external entities. Figure 5 illustrates the results of this experiments on 10-100 Twitter data. The times shown in each point of the chart are the median of the 10 collected data from the experiment. The results of this experiments on 100-500 data are presented with that of others in table II.

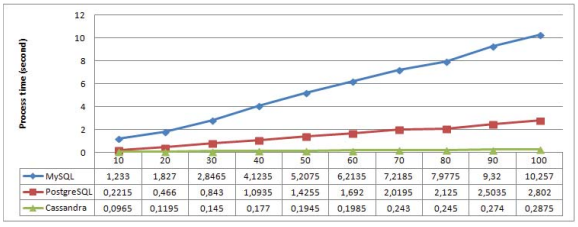

Fig. 5. Time for saving 10 - 100 Twitter data with the second condition

Similarly, the chart show that MySQL spent the longest time among the other two while Cassandra spent

the shortest write time. The time differences in this experiments are way smaller than that in the previous one. It indicates that the presence of Twitter external entities and flat-tabel does affect the write operation time. Moreover, the Mood median test results also show that the time difference are statistically significant.

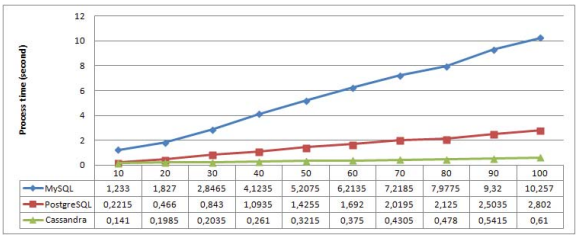

Fig. 6. Time for saving 10 - 100 Twitter data between SQL with first condition and Cassandra with second condition

Figure 6 presents the time comparison chart between SQL in the first experiments and Cassandra in the second experiment. It shows that even when SQL databases have the advantage, Cassandra still has the faster offered write operation time. The Mood median test results state that the differences are statistically significant.

As mentioned above, table II presents the write opera-tion experiment results for each databases and conditions for 100 until 500 Twitter data. The number next to the database name indicate the experiment number. The time results are presented as second.

TABLE II.
TIME FOR SAVING 100 - 500 TWITTER DATA (S)

Database Name	Number of Data				
	100	200	300	400	500
MySQL - 1	35.535	69.578	101.461	132.792	165.265
MySQL - 2	10.257	21.254	31.265	40.701	50.998
PostgreSQL - 1	9.407	19.102	27.782	36.297	45.445
PostgreSQL - 2	2.802	5.707	8.683	11.309	13.817
Cassandra - 1	0.610	1.174	1.669	2.151	2.677
Cassandra- 2	0.287	0.588	0.8415	1.080	1.314

In a different manner than before, the results of read operation experiment are presented in Table III. From the table, PostgreSQL has the upper hand when reading 5,000 Twitter data than MySQL. However, MySQL is faster on the other number. The Mood median test results for the SQL database resulting inconsistency which state that within 1,000 and 10,000 data the differences are significant while they are not within 5,000 data. Overall, MySQL still has the favor over PostgreSQL in read operation time.

TABLE III.
TIME FOR READING 1,000 - 10,000 TWITTER DATA (S)

Database Name	Number of Data		
	1,000	5,000	10,000
MySQL	0.000773	0.063045	0.099678
PostgreSQL	0.012750	0.057000	0.107950
Cassandra	0.005522	0.005866	0.006658

MySQL is then compared with Cassandra. Based on the Table III, Cassandra is slower when reading 1,000 Twitter data but its speed exceeds MySQL when reading more than 5,000 Twitter data. The Mood median test results state that MySQL is significantly faster among the other two while reading only 1,000 Twitter data while Cassandra is significantly faster when reading more than 5,000 data.

In summary, the results are somewhat relevant with the prior work [11]. At some point, Cassandra is not faster than several SQL databases in term of read operation. On the other hand, in term of write operation, Cassandra is faster than SQL databases.

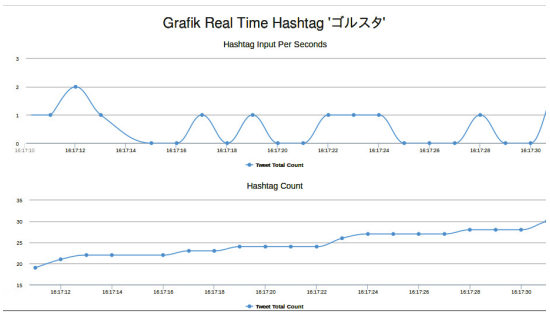

Fig. 7. Near real-time hashtag input and total count charts for certain hashtag

After the experiments are done, we build a web-based dashboard implementing Highcharts, a real-time charting JQuery library, in order to visualize it in near real-time. Figure 7 shows one of the real time chart implementation in showing hashtag input count and total count for selected hashtag. The chart provide the change in numbers of selected element in every seconds. However, the charts still have one seconds empty gap every 20 seconds due to the accumulation of process time between read and write operation from within the ETL process and the web-based dashboard.

V. CONCLUSIONS AND FUTURE WORK

This paper mainly focused on near real-time Twitter data warehouse development using Cassandra and its write and read operation comparison with that using

relational databases. According to the experiment results, even with the presence of massive data duplication, and the lack of aggregate and join operations, and even when the SQL databases has the advantages, Cassandra still provide a significantly better performance for near real-time data warehouse implementation. Although, Cassan-dra indeed is not the best when reading a relatively small data.

In overall, with the given configuration, DW architec-ture, and schema design, the near real-time Twitter data warehouse is successfully implemented in Cassandra with some drawbacks. First, the visualization still leaves one seconds gap every 20 seconds. Seconds, despite the high write operation, Cassandra could not do so when it comes to aggregate operation. Not to mention how we need to design the queries before designing the table.

We believe that there are many possibilities for future work. One possibility is to enhance the near real-time analytic side of this work and to cover the join and aggre-gate limitation by integrating Cassandra using real-time analytic framework such as Spark. Another possibility is to observe and improve the performance by developing it in a cluster, or even compare it with other NoSQL database such as HBase.

REFERENCES

[1] A. Sameh, "A Twitter Analytic Tool to Measure Opinion, Influence and Trust", in *Journal of Industrial and Intelligent Information*, vol. 1, no. 1, pp. 37-45, 2013.

[2] N. U. Rehman, S. Mansmann, A. Weiler, and M. H. Scholl, 2012, "Building a Data Warehouse for Twitter Stream Exploration", *in Proceedings of the 2012 IEEE/ACM International Confe- rence on Advances in Social Networks Analysis and Mining.*, Istanbul, 2012, pp. 1341-1348

[3] S. Wan, C. Paris, and D. Georgakopoulos, "Social Media Data Aggregation and Mining for Internet-Scale Customer Re-lationship Management", *in Proceedings of the 2015 IEEE 16th International Conference on Information Reuse and Integration.*, San Fransisco, 2015, pp. 39-48.

[4] D. M. Haughton, J. J. Xu, and D. J. Yates, "Introduction to Social Media and Data Analytics Minitrack", *in Proceedings of the 2013 46th Hawaii International Conference on System Sciences.*, Hawaii, 2013, pp. 1569.

[5] N. Bekmamedova, dan G. Shanks, "Social Media Analytics and Business Value: A Theoretical Framework and Case Study", *in Proceedings of the 2014 47th Hawaii International Conference on System Science.*, Hawaii, 2014, pp. 3728-3737.

[6] A. Bruns and S. Stieglitz, "Towards More Systematic Twitter Analysis: Metrics for Tweeting Activities", *International Journal of Social Research Methodology*, vol. 16, no.2, pp. 91-108, 2013.

[7] R. Burtica, E. M. Mocanu, M. I. Andreica, and N. Tapus, "Prac-tical Application and Evaluation of NoSQL Databases in Cloud Computing", *in Proceedings of Systems Conference (SysCon), 2012 IEEE International.*, Vancouver, 2012, pp. 1-6.

[8] M. S. Neethu and R. Rajasree, "Sentiment Analysis in Twitter using Machine Learning Techniques", *in Proceedings of the 2013 Fourth International Conference on Computing, Communica-tions and Networking Technologies (ICCCNT) on.*, Tiruchengode, 2013, pp. 1-5.

[9] L. Dehdouh, F. Bentayeb, O. Boussaid, and N. Kabachi, "Colum-nar NoSQL CUBE: Agregation Operator for Columnar NoSQL Data Warehouse", *in Proceedings of the 2014 IEEE Conference on Systems, Man, and Cybernatics.*, San Diego, 2014, pp. 3828-3833.

[10] R. Lawrence, "Integration and Virtualization of Relational SQL and NoSQL Systems including MySQL and MongoDB", *in Pro-ceedings of the 2014 International Conference on Computational Science and Computational Intelligence.*, Las Vegas, 2014, pp. 285-290.

[11] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases", *in Proceedings of the Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on.*, Victoria, 2013, pp. 15-19.

[12] J. Bhogal and I. Choksi, "Handling Big Data using NoSQL", in *Proceedings of the 2015 29th Interna- tional Conference on Advanced Information Networking and Applications Workshops.*, Gwangju, 2015, pp. 393-398.

[13] S. Kumar, F. Morstatter, and H. Liu, *Twitter Data Analytics*. Springer, 2013.

[14] E. Capriolo, *Cassandra High Performance Cookbook*. Birmingham, U.K. :Packt Publishing Ltd, 2011.

[15] J. Han, H. E, G. Le, and J. Du, "Survey on NoSQL Database", in *Proceedings of the Pervasive Computing and Applications (ICPCA), 2011 6th International Conference.* Port Elizabeth, 2011, pp. 363-366.

[16] A. Chebhotko, A. Kashlev, and S. Lu, "A Big Data Modeling Methodology for Apache Cassandra", *in Proceedings of 2015 IEEE International Congress on Big Data.* New York, 2015, pp. 238-245.

[17] L. Chen, W. Rahayu, and D. Taniar, "Towards Near Real-Time Data Warehousing", *in Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications.*, Perth, 2010, pp. 1150-1157.

[18] Y. Zhu, L. An, and S. Liu, "Data Updating and Query in Real-time Data Warehouse System", *in Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, Wuhan, pp. 1295-1297.

[19] C. Wang and S. Liu, "SOA Based Electric Power Realtime Data Warehouse", *in Proceedings of the 2008 Workshop on Power Electronics and Intelligent Transportation System.*, Guangzhou, 2008, pp. 355-359.

[20] A. C. Carniel, A. de Aguiar S ; V. H. P. Brisighello ; M. X. Ribeiro ; R. Bueno ; R. R. Ciferri ; C. D. de Aguiar Ciferri, "Query processing over data warehouse using relational databases and NoSQL", Informatica (CLEI), 2012, *XXXVIII Conferencia Latinoamericana En*, Medellin

[21] N. U. Rehman, S. Mansmann, A. Weiler, and M. H. Scholl, 2012, "OLAPing Social Media: The case of Twitter", *in Pro-ceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.*, Niagara Falls, 2013, pp. 1139 - 1146

[22] H. H. Lemmer, "Adaptive Test for the Median", *IEEE Trans-action on Reliability.*, vol. 3, no. 42, pp. 442-448, 1993.