ISSN 2348-1218 (print)
International Journal of Interdisciplinary Research and Innovations ISSN 2348-1226 (online)
Vol. 7, Issue 3, pp: (15-23), Month: July - September 2019, Available at: www.researchpublish.com

A Performance Comparison between Modern Relational Database VoltDB and Graph NoSQL Database Neo4j

Khawlah A. Altuwairqi[1], Dr. Fahad Algurashi[2]

Department of Computer Science, Faculty of Computing and Information Technology, FCIT King Abdul-Aziz University, KAU Jeddah, Saudi Arabia

khawlah.altuwairqi@gmail.com[1], fahad@kau.edu.sa[2]

Abstract: **Since the 1980s, relational database management systems have taken precedence over other data models (hierarchical model and network). Appreciated by companies for its highly coherent data structure and ACID transaction support, it is now being challenged by the emergence of new needs linked to the emergence of Big Data. This phenomenon implies that some companies must now manage huge volumes of data in exponential growths. Responding to this problem, the NoSQL appears as a viable solution but not without defects. Moreover, it has the advantage of being available in several types of bases more adapted to the specific needs of companies. However, the newcomer "NewSQL" seems to promise an architecture combining the advantages of the relational model and the NoSQL. Most of recently works were compared between OldSQL with graph NoSQL, or OldSQL with NewSQL. Some of them compared between all three types but without experimental result. In this paper, discusses SQL, graph NoSQL and NewSQL. Graph Neo4j and VoltDB were selected in from various open source NoSQL and NewSQL. Characteristics and comparisons performance of each one was studied. CRUD operations were applied in VoltDB and Neo4j with varying datasets size. As a result, VoltDB is significantly better than Neo4j insert, update and delete. Neo4j is significantly better than VoltDB in select.**

Keywords: **NoSQL, NewSQL, Big Data, ACID properties, BASE properties, CRUD, Neo4j, VoltDB.**

1. INTRODUCTION

Since the 1980s, relational database management systems have become increasingly important to other data management systems. Today, still used by most companies they are always appreciated for their capacities to ensure a high consistency of the data and guarantee a reliability during the transactions [1]. However, the emergence of decision-making systems and the explosion of data volumes have led many companies to de-standardize their data model. This technique for grouping information into aggregates aims to optimize response times by breaking with the three normal forms so dear to the RDBMS [2].

The Big Data led the big Internet players (Google, Facebook, and Amazon) to develop and then adopt alternative technologies named NoSQL [3]. These allow them to support a horizontal loading while ensuring a flexibility of the data model. From then on, NoSQL is a solution for the company wanting to manage high loads and volumes [2]. However, this technology sacrifices design for consistency to the benefit of availability. In this model, ACID properties are often set aside for performance [4]. In addition, the flexibility offered by the without schema and the abandonment of SQL make it a flexible technology and particularly appreciated by developers. They discover a DBMS where the application becomes master of the schema of the database. More endless disputes with a DBA that imposes a non-flexible database schema [3].

The difficulty in managing the low data consistency for developers led the great tenors of the web to develop NewSQL [5]. This new RDBMS allows for horizontal scalability, schema flexibility and high data consistency through ACID transactions. The NewSQL is as young as it is full of promises. It does not have feedback from RDBMS and NoSQL [6].

The paper's structure is as follows. Section 2 presents the classification and big data in traditional database on brief. Non-relational and modern relational database are discussed in section 3 and 4 respectively. In section 5, reviewed of related works. Characteristic of selected databases and Experiment setup are discussed in section 6 and 7 respectively. Result and analysis are discussed in section 8. Finally, section 9 and 10 concludes the paper and features work respectively.

2. THE RELATIONAL (TRADITIONAL) MODEL - SQL

The growing use of computing in large enterprises led to the need for a well-structured data model that separated logical representation from data and their physical organization. The relational data model allows the independence between the physical structure of the files containing the data and their logical organization [7].

In compliance with the normal forms, it offers a high degree of data consistency and support for ACID transactions. We will explain these last notions in the following sections. In the following part, the essential characteristics of the relational model will be explained [8].

2.1 Characteristics of SQL

2.1.1 Representation. The data is thus represented in two-dimensional "table" objects. The columns (attributes) represent the values of a particular type for a particular domain (e.g. "name" column of type varchar2). The rows contain the records (tuples) of the table [4]. Each of them has an identifier that guarantees its uniqueness.

2.1.2 Language. The database management systems supporting the relational model (RDBMS) use the structured query language "SQL" to act on the data [1]. Using an easy-to-access syntax, you can perform CRUD operations on the data. CRUD stands for Create, Read, Update and Delete [2, 9].

2.1.3 ACID Transaction. Support of ACID transactions is strengths in relational model. ACID stands for Atomicity, Consistency, Insulation and Durability [10]. The atomicity property ensures that a transaction is complete or not at all. If a part of a transaction cannot be made, all traces of the transaction must be erased and the data must be returned to the state in which it was Before the transaction. The consistency property ensures that each transaction will bring the system from a valid state to another valid state [8]. The isolation property ensures that the simultaneous execution of transactions produces the same state that would be obtained by the serial execution of the transactions. No possible dependency between transactions [9]. The durability property ensures that when a transaction has been confirmed, it remains registered even if any problem appear. The respect of the ACID during a transaction, is a guarantee of its reliability [4].

2.2. SQL in Big Data

The Big Data is a word to explain the explosion of the volumes of data. It is described by three terms: Volume, Velocity and Variety. The volume of data grows exponentially. Big players like Facebook, Google and Amazon were the first to see this growth. The velocity of data flow. They are continuous. They must be collected and analyzed in real time. The variety of data types, such as semi-structured types (e.g. XML and JSON) and unstructured (e.g., photos and videos) are constantly increasing due to the new architectures used [11].

The exponential increase in bandwidth facilitated by the reduction in the number of machines and the democratization of high-speed lines are a new challenge for many companies. The relational model seems increasingly "outdated" to handle these masses of data. Relational databases are unable to manage efficient horizontal scaling on large volumes of data [8].

3. THE NON-RELATIONAL MODEL – NOSQL

In response to the emergence of Big Data, the big Internet companies (Google, Facebook, Amazon etc.) each developed alternatives to SQL. NoSQL database were developed to meet the needs generated by Big Data. It referred to as Not-only-SQL databases. The interest of NoSQL storage systems lies mainly in the software architecture choices that have been taken during their designs [12]. Among the main reasons that led to the creation of these systems are two main points [13]:

- The possibility of using something other than a fixed schema in the form of tables with all the properties fixed in advance.
- The ability to have a system easily distributed across multiple servers and with which an additional need for storage or scalability simply means adding new servers.

NoSQL databases are respond correctly to the challenges caused by the explosion of data volumes [12]. In the following part, the type of NoSQL will be explained.

3.1 Types of NoSQL

There are several types of NoSQL databases. However, they all have one thing in common, it is the abandonment of constraints close to the data. These databases store either key-value pairs or JSON27 documents [11]. The developer is free to store and organize the data as he wants in base. It is he who defines through his application the structure and rules of the elements that will be manipulated and saved [14].

3.1.1. Key-Value Stores. It is the simplest model of the different basic types of NoSQL. It has a significant advantage, it is easily scalable horizontally. When adding a new node, it will be sufficient to redefine the key intervals for which the cluster servers are responsible [14].

3.1.2. Column-oriented Stores. Column-oriented databases work by column families. In this type, data is no longer stored on two dimensions (row X column) but on column only. This reduces the time and random access to the disk [12].

3.1.3. Document Stores. Document-oriented databases approach the key-value model. The value being this time characterized by a document in a hierarchical format whose structure is free. It is identified by a unique key. This type of model allows non-planar data to be stored. This is particularly suitable for web applications [13].

3.1.4. Graph Databases. The NoSQL graphic-oriented databases seek above all to answer complex or even impossible problems for RDBMS. NoSQL-based graph- based databases are more efficient than their relational rivals for large volumes of data. Graph databases are databases that used graph form to store data. It consists of nodes act as the (objects) and edges act as (relationship) between the objects [15]. This type of database is very flexible because it does not impose a fixed data scheme as is the case with a RDBMS that respects normal forms. In addition, it allows to run graph algorithms regularly used in a highly-connected data model. It uses the cypher query language "CQL" to act on the data [16]. In the following part, essential characteristics of the NoSQL will be explained.

3.2. Characteristics of NoSQL

3.2.1. CAP theory. While the relational model obeys normal forms and ACID rules, the NoSQL databases follow CAP theorem. CAP stands for Consistency, Availability and Partition Tolerance. CAP have 3 principles [4]. In Consistency, all clients see the same view even when there are upgrades, in fact it is the 'Atomic' of the ACID properties. In Availability, all customers can find replicated data even when damage occurs. In Tolerant to partition, the system is tolerant to partitioning, mean that the data can be distributed on several machines. NoSQL databases are bound to respect two out of three principles. They will choose one of the three cases according to the needs to be filled:

- CA: Consistency and high availability.
- CP: Consistency and partition tolerance.
- AP: High availability and partition tolerance. Often the need for availability and partitioning is more important than the consistency. BASE propriety relay on AP, and ACID propriety relay on CA. NoSQL used BASE propriety. BASE stands for Basically Available, Soft state and Eventually consistent [6].

3.3. Shortcoming in NoSQL

In view of the above, one might thus think that the NoSQL presents only advantages. However, the virtual disregard of high data consistency for performance and high availability is a disadvantage of this technology [8]. Ensuring consistent data in a redundant system often has a major blow. It will be necessary to wait until all the replicas

containing the identical data on which are operated are days before a transaction on the system. Therefore, in most cases it is accepted that the data may be coherent. This of course can only be applied if we agree to work with data that is not always up to date [14].

4. THE MODERN RELATIONAL DATA MODEL- NEWSQL

In the previous sections, we talked about the new needs that have arisen since the emergence of Big Data and NoSQL databases, which may in some cases be a relevant answer. However, it appears that this technology sacrifices most of the time the high degree of data consistency so dear to the relational model, benefiting from high availability and horizontal scalability. On the other hand, it does not suit everyone [6].

Some companies simply can't do without ACID transactions. Moreover, the developer will be led to "juggle" with the possible consistency of the data. In addition, each NoSQL database management system comes with its own query language. There is thus a lack of standardization of the interfaces between the applications. NewSQL database gathering the advantages of NoSQL and RDBMS [5].

Since then, several implementations of this new type of architecture have appeared very recently. It uses the structured query language "SQL" to act on the data [5]. In the following part, the characteristics of NewSQL will be explained.

4.1 Characteristics of NewSQL

He is as young as he is full of promises. Below are some of these features [6]:

- SQL as common query language and ACID Transaction
- A mechanism that avoids pause of locks during concurrent read operations with write operations. This makes it easier to read in real time.
- An architecture that has better performance per node than conventional RDBMS solutions.
- Horizontal scalability, architecture without master and able to turn on many knots without suffering bottleneck [17].

In view of the above, it can reasonably be expected that this type of architecture will be the one adopted tomorrow in company. In the following section, the recently related works were studied.

5. RELATED WORKS

In [9], the authors were compare the performance of MySQL, MongoDB and Neo4j. They were applied 7 queries. In queries find rating by id user, count movie rate, find movies that have high reviews with highest score and find the most popular movies from all users, the MongoDB was the faster compared with others. In query find movies rate by user id, the MySQL was the better. In queries find movie by id and calculate average of movies score, the Neo4j was better than others.

In [10], the authors were proposed system allow the user to select database and perform transaction on it. They were compare the performance of MySQL, MongoDB, VoltDB and FoundationDB. The FoundationDB was better performance than other three databases. In [17], the authors compare the performance of MySQL, SQLite, MongoDB and VoltDB. They were applied four operations insert, select, update and delete. VoltDB performance in four operation was better than MongoDB and MySQL. SQLite performance in select, update and delete was better than all others. VoltDB performance in insert operation was better than all others.

In [18], the authors were compare the performance of Neo4j graph database and MySQL relational database. They were used integer payload databases and character payload databases with varying size. MySQL was better speed in integer database and Neo4j was better speed in character database. In other hand. this paper [19] was also compare the performance of Neo4j and MySQL. This comparison was applied in social networking field. They were used 200 users to selected from database. Neo4j was more speed than MySQL. Neo4j took 3 seconds while MySQL took 152 seconds.

In [20], the authors compare the performance of Column NoSQL HBase and Neo4j. They applied four groups of queries. In simple aggregate queries, the HBase was outperform Neo4j. In range queries, the performance of HBase very close to Neo4j. In top-k queries and group-by, the Neo4j was outperform HBase. Most of recently works were

compared between OldSQL with graph NoSQL, or OldSQL with NewSQL. Some of them compared between all three types but without experimental results. Contribution in this work is compare between NewSQL and Graph NoSQL in performance of runtime. The researcher selected Neo4j and VoltDB. Neo4j and VoltDB are one of the main graph NoSQL and NewSQL databases respectively. In the following part, the characteristics of Neo4j and VoltDB will be explained.

6. DATABASE SELECTED

Neo4j and VoltDB were selected in this study for some characteristic as the following:

Neo4j version 3.1.3 is used to represent experiment result of graph NoSQL database. It is open source for all noncommercial uses. World's First and Best Graph Database. It has the largest and most vibrant community. It provides a high-performance in read and write. In addition, it provides integrity of data and protecting. It's easy to learn and understand. Its provide a simple user interface. its useful in social network [21].

VoltDB version 7.1.1 is the implementation chosen to represent NewSQL databases. VoltDB was built to do important things better than any other system: ingest millions of data points from millions of users, devices and sensors; perform real-time analytics; and act in milliseconds – with 100% correct data – millions of times a day. In the following part, datasets and experiment setup will be explained [22].

7. DATASETS AND EXPERIMENT SETUP

Datasets provided by Chicago Police Department [23], it shown in following table1

Table 1: Datasets Information

Dataset	Table Name	Record Number	Size
1	crime1	68,075	16.1MB
2	crime2	486,739	112.9MB
3	crime3	3,056,540	722.2MB

Hardware and software configurations were as the following:

Model	MacBookPro (13-inch, 2016)
Processor	2.9 GHz Intel Core i5
Operating System	MacOS Sierra 10.12.4
RAM	8 GB 2133 MHz LPDDR3
Disk	512 GB Macintosh HD
VoltDB	Version 7.1.1
Neo4j	Version 3.1.3

8. RESULTS AND ANALYSIS

In experiment results, the CRUD operations were applied in VoltDB and Neo4j with varying datasets size as we mentioned. CRUD is Create(insert), Read(select), Update and Delete. The runtime average of 5 were taken. All times are in seconds (s).

8.1 Insert (Create)

In this operation, different datasets size was inserted into the databases. Insert record by write the following code in terminal:

VoltDB:

/csvloader crime -f crime.csv

Neo4j:

/neo4j-import --into crime --nodes crime.csv

ISSN 2348-1218 (print)
International Journal of Interdisciplinary Research and Innovations ISSN 2348-1226 (online)
Vol. 7, Issue 3, pp: (15-23), Month: July - September 2019, Available at: www.researchpublish.com

Table 2: Results of Insert Operation

Dataset	VoltDB speed	Neo4j Speed
1	1.8436s	3.5596s
2	5.1146s	12.7598s
3	31.0736s	60.811s

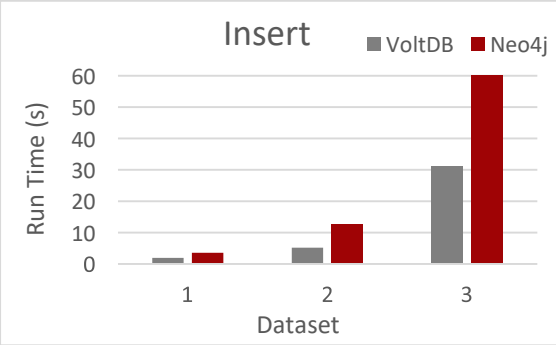

Figure 1: Insert Results of VoltDB and Neo4j

In table2, the result of insert operation shown. When the dataset was increase the insert of runtime was increase. It can be seen VoltDB is significantly better than Neo4j on the insert operation performance as the figure1 shown. Neo4j was take more than double time of VoltDB.

8.2 Select (Read)

In this operation, selected and return primary_type and description in each dataset. The code is as the following:

VoltDB:

SELECT primary_type, description FROM Crime

Neo4j:

MATCH (n :Crime)

RETURN n.PRIMARY_TYPE, n.DESCRIPTION;

Table 3: Results of Select Operation

Dataset	VoltDB speed	Neo4j Speed
1	5.837s	0.2704s
2	47.4978s	1.4978s
3	119.5356s	10.9126s

Figure 2: Select Results of VoltDB and Neo4j

In table3, the result of select operation shown. When the dataset was increase the select of runtime was increase. It can be seen Neo4j is significantly better than VoltDB on the select operation performance as the figure2 shown. VoltDB was take long time in select.

8.3 Update

In this operation, updated all record and set primary_type value "theft" in each dataset. The code is as the following:

VoltDB:

UPDATE Crime

SET primary_type = 'theft'

Neo4j:

MATCH (n :Crime)

SET n.PRIMARY_TYPE = "theft"

Table 4: Results of Update Operation

Dataset	VoltDB speed	Neo4j Speed
1	0.1204s	0.2762s
2	0.7474s	2.1684s
3	5.087s	9.5836s

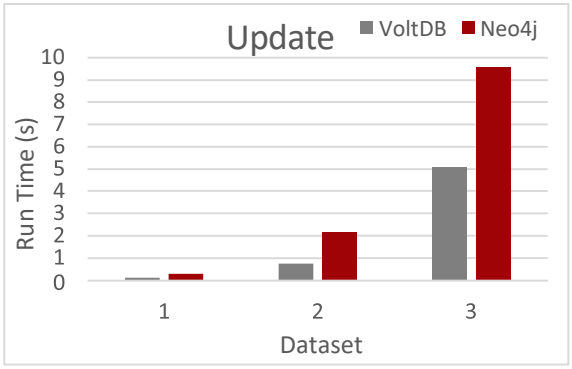

Figure 3: Update Results of VoltDB and Neo4j

In table4, the result of update operation shown. When the dataset was increase the update of runtime was increase. It can be seen VoltDB is significantly better than Neo4j on the update operation performance as the figure3 shown. Neo4j was take more than double time of VoltDB.

8.4 Delete

In this operation, deleted all records in each dataset.

The code is as the following:

VoltDB:

DELETE

FROM Crime

Neo4j:

MATCH (n :Crime) DELETE n

Table 5: Results of Delete Operation

Dataset	VoltDB speed	Neo4j Speed
1	0.061s	0.6078s
2	0.0658s	2.2594s
3	0.0952s	4.4198s

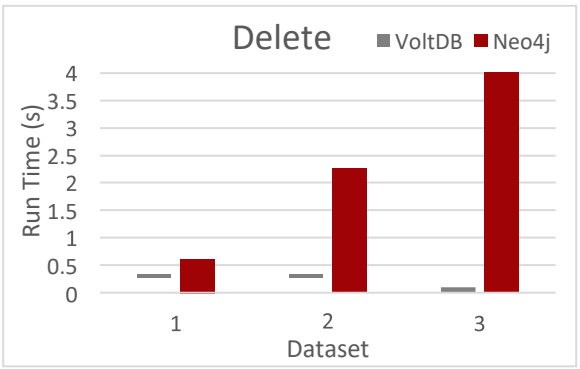

Figure 4: Delete Results of VoltDB and Neo4j

In table5, the result of delete operation shown. When the dataset was increase the delete of runtime was increase. It can be seen VoltDB is significantly better than Neo4j on the delete operation performance as the figure4 shown. Neo4j was take more than double time of VoltDB.

9. CONCLUSION

Since emergence of Big Data, SQL are unable to manage efficient horizontal scaling on large volumes of data. Responding to this problem, the NoSQL appears. It has the advantage of being available in several types of bases more adapted to the specific needs of companies. In other hand, NoSQL doesn't support ACIDTransaction.

Wherefore, the newcomer "NewSQL" seems to promise an architecture combining the advantages of the relational model and the NoSQL. In this paper, discusses SQL, graph NoSQL and NewSQL. Graph Neo4j and VoltDB were selected in from various open source NoSQL and NewSQL. Characteristics and comparisons performance of each one was studied. As experiment results, CRUD operations were applied in VoltDB and Neo4j with varying datasets size as we mentioned. CRUD is Create(insert), Read(select), Update and Delete. The runtime average of 5 were taken.

As a result, it can be seen VoltDB is significantly better than Neo4j on the insert, update and delete operations performance. In addition, Neo4j is significantly better than Neo4j on the select operation performance.

10. FUTURE WORKS

The author will apply different techniques to reduce run time in each operation. In other hand, different database in graph and NewSQL will used.

REFERENCES

[1] Yassien, A.W. and Desouky, A.F., "May. RDBMS, NoSQL, Hadoop: A Performance-Based Empirical Analysis", *In Proceedings of the 2nd Africa and Middle East Conference on Software Engineering*, ACM, 2016, pp. 52-59.

[2] C. Győrödi, R. Győrödi, G. Pecherle, and A. Olah, "A comparative study: MongoDB vs. MySQL", *2015 13th International Conference on In Engineering of Modern Electric Systems (EMES)*, IEEE, 2015, pp. 1-6.

[3] C. Győrödi, R. Győrödi, and R. Sotoc, "A Comparative Study of Relational and Non-Relational Database Models in a Web-Based Application", *International Journal of Advanced Computer Science and Applications*, 2015, pp.78-83.

[4] S. Pore, and S. Pawar, "Comparative Study of SQL & NoSQL Databases", *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2015, pp. 1747-1753.

[5] H. Fatima, and k. Wasnik, "Comparison of SQL, NOSQL and New SQL Databases in Light of Internet of Things – A Survey", *Proceedings of 48th IRF International Conference*, 2016, pp. 1- 4.

[6] S. Binani, A. Gutti, and S. Upadhyay, "SQL vs. NoSQL vs. NewSQL-A Comparative Study", *Communications on Applied Electronics (CAE)*, Foundation of Computer Science FCS, New York, USA, 2016, pp. 43-46.

[7] T. C. Hsu, D. M. Chang and H. J. Lee, "The Study of Application and Evaluation with NoSQL Databases in Cloud Computing," *2014 International Conference on Trustworthy Systems and their Applications*, Taichung, 2014, pp. 57-62.

[8] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, "A survey and comparison of relational and non-relational database", *International Journal of Engineering Research & Technology*, 2012, pp. 1-5.

[9] E. Chavez, and M. Moraga, "Linked data performance in different databases Comparison between SQL and NoSQL database", thesis, 2014.

[10] M. Katkar, "Performance Analysis for NoSQL and SQL", *International Journal of Innovative and Emerging Research in Engineering*, 2015, pp.12-17.

[11] Z.H. Liu, B. Hammerschmidt, D. McMahon, Y. Liu, andH.J. Chang, "Closing the functional and performance gap between SQL and NoSQL*", In Proceedings of the 2016 International Conference on Management of Data*, ACM, 2016, pp. 227-238.

[12] R. Hecht, and S. Jablonski, "NoSQL evaluation: A use case oriented survey", *2011 International Conference on Cloud and Service Computing (CSC)*, IEEE, 2011, pp. 336-341.

[13] A. Bansel, H. González-Vélez and A. E. Chis, "Cloud- Based NoSQL Data Migration," *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, Heraklion, 2016, pp. 224-231.

[14] A. Nayak, A. Poriya, and D. Poojary, "Type of NOSQL databases and its comparison with relational databases", *International Journal of Applied InformationSystems*, Foundation of Computer Science FCS, 2013, pp.16-19.

[15] S. Patil, G. Vaswani and A. Bhatia, "Graph Databases- An Overview", *International Journal of Computer Science and Information Technologies*, 2014, pp. 657- 660.

[16] R. Angles, "A Comparison of Current Graph Database Models," *2012 IEEE 28th International Conference on Data Engineering Workshops*, Arlington, VA, 2012, pp. 171-177.

[17] Y. Wang, G. Zhong, L. Kun, L. Wang, H. Kai, F. Guo, Liu, and X. Dong, "The Performance Survey of in Memory Database", *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, IEEE, 2015, pp. 815-820.

[18] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, andWilkins, "A comparison of a graph database and a relational database: a data provenance perspective", *In Proceedings of the 48th annual Southeast regional conference*, ACM, 2010, pp. 1-42.

[19] R. Fatima, A. Ahmed, "Role of Graph Databases in Social Networking Sites: A Performance Comparison between Graph Database Neo4j and Relational Database Mysql in Social Networking Sites", *Journal of Independent Studies and Research – Computing*, 2012, pp. 22-25.

[20] M. Kendea, V. Gkantouna, A. Rapti, S. Sioutas, G. Tzimas, and D. Tsolis, " Graph DBs vs. Column- Oriented Stores: A Pure Performance Comparison", *In Algorithmic Aspects of Cloud Computing*, Springer International Publishing, 2016, pp. 62-74.

[21] https://neo4j.com

[22] https://www.voltdb.com

[23] https://portal.chicagopolice.org