Performance Analysis of Neo4j and MySQL Databases using Public Policies Decision Making Data

Rahmatian Jayanty Sholichah
Telkom University
School of Computing
Bandung, Indonesia
jayrahma@student.telkomuniversity.ac.id

Mahmud Imrona
Telkom University
School of Computing
Bandung, Indonesia
mahmudimrona@telkomuniversity.ac.id

Andry Alamsyah
Telkom University
School of Economic and Business
Bandung, Indonesia
andya@telkomuniversity.ac.id

***Abstract*—Currently, the development of data has increased rapidly, Solutions are needed to be able to manage data efficiently, one that can be offered is to utilize the database. The biggest decision in selecting a database is to select between SQL or NoSQL. MySQL is a database that uses SQL as a query language, consists of tables that store data in the form of columns and rows, then the new format database NoSQL, appeared, it is suitable for handling large amounts of data in a variety of formats. Neo4j is one of NoSQL that is widely used, it is a graph database which provides an easy way to visualize data by storing data in the form of nodes that are connected by edges. In this paper, we compared the performance of MySQL and Neo4j databases in terms of memory usage and execution time, also we presented the flexibility of the databases using P. The results show that MySQL has a faster execution time than Neo4j, although, both these databases have the same time complexity. It is also known that Neo4j has a higher memory usage than MySQL. But Neo4j has better flexibility than MySQL.**

Keywords*—*data, database, performance, mysql, neo4j

I. Introduction

In this era, the development of data has increased rapidly, these data have been stored digitally, so that later we can obtain information in an easier way [1][2]. However, to be able to summarize and convert data into information we need a solution [3]. One that can be offered is to utilize the database, it is able to manage data effectively and can be used according to user requirements. Including data on public policies proposed by Deputies, this solution needs to be implemented so that the data is processed into information that can help in making decisions.

The biggest decision in database selection is to select between SQL or NoSQL. Both have their own characteristics so users must determine their requirements. Database selection is caused by its performance, which can be partly measured by the memory usage, the execution time and the flexibility.

MySQL is a database that uses SQL as a query language. It stores data in tables, consisting of rows and columns [4] Meanwhile, Neo4j is one of NoSQL that is widely used [5]. It is a graph database which is suitable for processing massive amounts of data by visualizing it in the form of nodes and edges [6][7].

In this paper, we compared the performance of MySQL and Neo4j databases in terms of memory usage and execution time, because the database processes data in memory, so it is necessary to declare memory usage, especially if the data being processed is large. Meanwhile, the query execution time is a parameter used to know the access speed from the databases [8]. Then we also presented the flexibility of the databases, so we know which one is easier to use for Developers, especially for handling big data, connected data, and data that continues to grow

II. Literature Review

A. Definition of SQL and NoSQL

SQL databases are designed to handle structured data, it processes data in a fixed format [1]. SQL databases are vertically scalable, it means that to handle large amounts of data, users must increase the capacity of systems such as CPU and RAM [4].

NOSQL databases (Not Only SQL) are databases designed to handle unstructured data, it is suitable for handling data with varying or non-fixed formats. NoSQL databases are horizontally scalable which involved adding more servers [1] [9].

B. MySQL

MySQL is a relational database management system that uses SQL as a query language, it has several data types which is declared when creating tables. The data type affects for the data in the table. Therefore, for all the data that will be enter to the table it must match with the declared data type.

The main command of MySQL is SELECT statement, it is a command used to display certain data from a table also usually used with join [10].

C. Neo4j

Neo4j is one of NoSQL, developed by Neo Technology, Inc. [11] It is a graph database that visualizes data in the form of nodes connected by edges. Node is a representation of objects. While edges are lines that connect between nodes and show relationships, which can have properties [7].

In Neo4j the data is taken in the form of a query and written in the Cypher language, it is a language that is designed to support various versions of pattern in data, making it a simple and logical language to learn. [12].

III. Research Methodology

A. Modelling Flow Description

The figure below is the modeling flow used in this research.

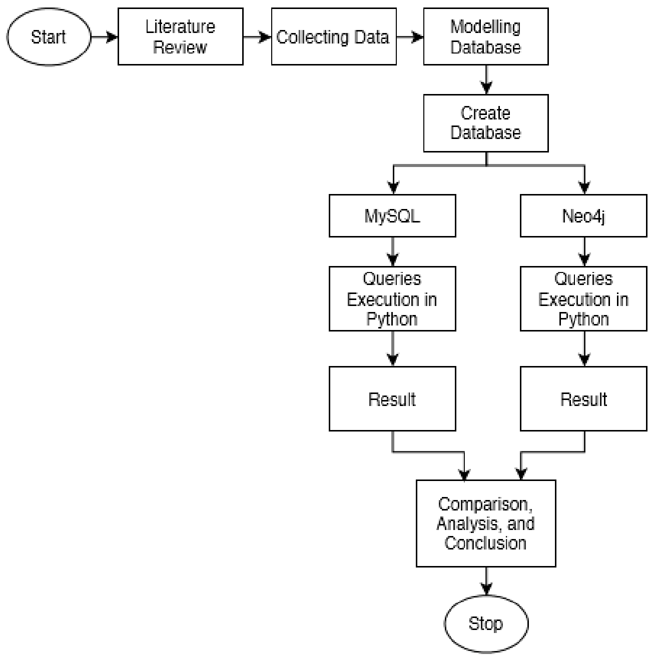


Fig. 1 Research Modeling Flow

1) Literature Review: Started by searching and reviewing literatures regarding the comparison of SQL (MySQL) and NoSQL (Neo4j) which is used as a guide in this research. The sources of information used include: Published scientific journals and Articles from certain web pages.

2) Collecting Data: After reviewing the literature, we searched and collected data, the dataset used in this research is data on public policies decision making by the Deputies in Germany, which data was obtained from github belonging to "isnok" [13]. The deputies here mean members of parliament in Germany who have an important role in making policies and laws.

3) Modelling Databases: At this stage the authors has made modeling for both databases. For MySQL the model used is a relation schema. Meanwhile, Neo4j uses the Graph schema.

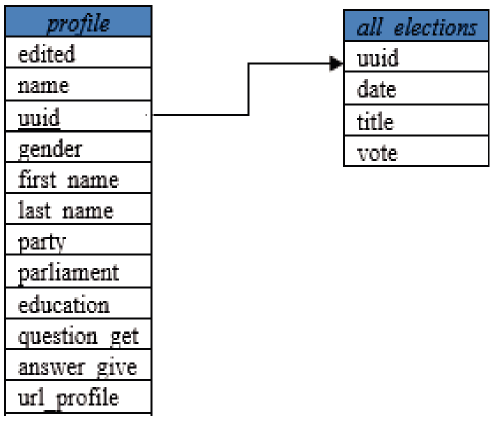


Fig. 2 Relation Schema in MySQL

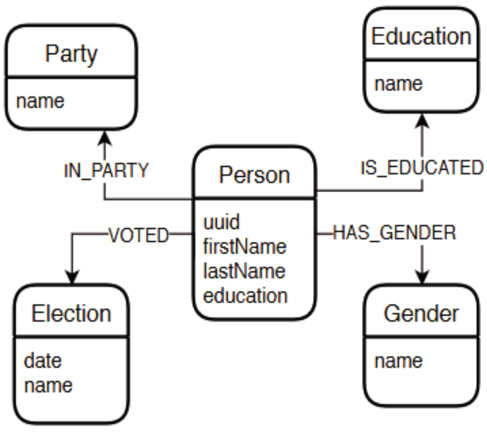


Fig. 3 Graph Schema in Neo4j

Fig. 2 shows that there are 2 tables, namely profile which has 12 keys and all_elections which has 4 keys. These two tables are connected by uuid attribute as the primary key in profile and become foreign key in all_elections.

Fig. 3 shows that there are 5 nodes with some labels, Person has 4 property keys such as uuid, firstName, lastName, and education. Gender with 1 property key that is name. Party with 1 property key that is name. Education with 1 property key, that is name and the last, Election with 2 property keys namely date and name. All nodes are associated by edge or relationship with relationship types:

- Person with Party: IS_PARTY
- Person with Education: IS_EDUCATED
- Person with Gender: HAS_GENDER
- Person with Election: VOTED

From Fig. 2 and Fig. 4, as we can see that connecting tables in the MySQL requires a primary key and a foreign key. Meanwhile, for Neo4j, it uses edge to represent the relationship between nodes.

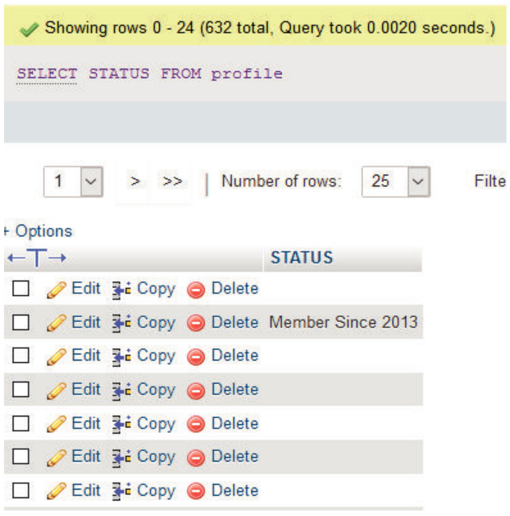
4) Create Databases: This stage began by building database systems, for MySQL the authors used phpMyAdmin and for Neo4j the authors used Neo4j Desktop. Then we loaded the dataset into both databases. After that, we connected the two databases with python, so we can get the result in the form of the execution time and the pid. From this pid we can find out the memory usage in the task manager, in the section of memory (Active Private Working Set) which is contained in the details, the private working set is the size of the working set which shows how much the process actually need and it cannot be shared among other processes [14]. In this stage we has added 1 data (Status), the aim is to show the flexibility of the databases. In MySQL, the first thing to do is adding 1 column (Status):

ALTER TABLE profile ADD Status VARCHAR (200) NOT NULL;

Then we start adding data to the column by updating the row:

UPDATE profile SET Status = 'Member Since 2013' WHERE profile.uuid = '014490fc-3403-4255-b92d - 175816269937';

However, this causes some rows in Status becomes empty.



```
Showing rows 0 - 24 (632 total, Query took 0.0020 seconds.)
SELECT STATUS FROM profile
```

	STATUS
Edit Copy Delete	
Edit Copy Delete	Member Since 2013
Edit Copy Delete	
Edit Copy Delete	
Edit Copy Delete	
Edit Copy Delete	
Edit Copy Delete	

Fig. 4 Status in MySQL

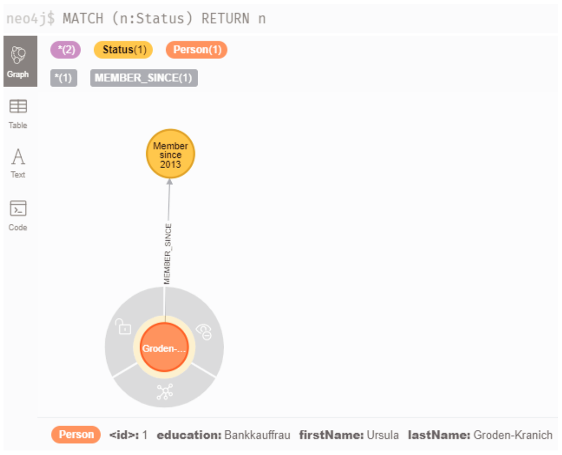
While in Neo4j, the first thing to do is creating Node:

CREATE (s:Status {status: 'Member since 2013'}) RETURN s;

After that, we can add the relationship of that node with other node:

MATCH (p:Person {uuid: '014490fc-3403-4255-b92d-175816269937'})
MATCH (s:Status {status: 'Member since 2013'})
CREATE (p)-[rel:MEMBER_SINCE]->(s);

Unlike MySQL, Neo4j can easily update data without changing other data structures.



```
neo4j$ MATCH (n:Status) RETURN n
Person <id>: 1 education: Bankkauffrau firstName: Ursula lastName: Groden-Kranich
```

Fig. 5 Status in Neo4j

5) Queries Execution in Python: There are 4 queries, 3 queries were executed with some limits like limit 10, limit 100, limit 500, limit 1000 and limit 10000 and the last query executed to find out the database's ability to handle unstructured data. Table I. below presents the queries and the descriptions.

TABLE I. TESTED QUERIES

Q	MySQL	Neo4j	Description	Simple Query	Join Query
1	SELECT title FROM all_elections WHERE vote='enthalten'	MATCH (p:Person)-[v:VOTED]->(e:Election) WHERE v.vote='enthalten' RETURN e.name	Policies, which decisions are "enthalten"	V	
2	SELECT last_name, title FROM profile JOIN all_elections ON (profile.uuid = all_elections.uuid) WHERE gender='female' and vote='NEIN'	MATCH (p:Person)<-[g:HAS_GENDER]-(gender:Gender) MATCH (p)-[v:VOTED]->(e:Election) WHERE gender.name = 'female' and v.vote='NEIN' RETURN p.lastName, e.name	Last name and policies rejected by female deputies		V
3	SELECT last_name, title, vote FROM profile JOIN all_elections ON (profile.uuid = all_elections.uuid) WHERE gender='male'	MATCH (p:Person)<-[g:HAS_GENDER]-(gender:Gender) MATCH (p)-[v:VOTED]->(e:Election) WHERE gender.name = 'male' RETURN p.lastName, e.name, v.vote	Last name, policies and decisions of male deputies		V
4	SELECT first_name, last_name, status FROM profile limit 5	MATCH (p:Person)-[rel:MEMBER_SINCE]->(n:Status) RETURN p.firstName, p.lastName, n.status limit 5	First name, last name, and status of the deputies	V	

6) Comparison, Analysis and Conclusion: At this stage we compared and analyzed the results of the query execution that has been obtained, for the execution time, the authors uses Big O Notation. Big O notation is a notation that describes the complexity of the algorithm with algebraic terms [15]. Big O notation itself is divided into several categories [16]:

- O(1): The Execution time is always constant, regardless of the number of n.
- O($\log n$): The Execution time grows logarithmically proportional to the number of n.
- O(n): The Execution time grows directly proportional to the number of n.
- O($n \log n$): The Execution time grows proportionately the number of n.

- O(n^c): The execution time grows faster than before based on the number of n.
- O(c^n): The Execution time grows faster than polynomial algorithms based on the number of n.
- O($n!$): The execution time grows fastest and unusable even on the smallest n.

According to [15] it is known that an algorithm can be the best if it has the notation O (1). It is said that it is good if the notation is O($\log n$). It is said to be quite good if it has O(n) notation. It is said to be bad if the notation is O($n \log n$) and it is said to be the worst if the notation is O(n^c), O(c^n), O($n!$). n here is the number of records displayed. The table below show comparison in Big O Notation based on the number of records.

TABLE II. BIG O COMPLEXITY COMPARISON

Number of Records	O(1)	O(log n)	O(n)
10	1	1	10
100	1	2	100
500	1	2.67	500
1000	1	3	1000
10000	1	4	10000

IV. RESULT

The table and graphics below show the comparison of the queries execution times and memory usage of the both databases.

TABLE III. THE RESULTS OF EXECUTION TIME AND MEMORY USAGE IN MYSQL AND NEO4J

Number of Records	Database	Time (s)			Memory (K)		
		Q1	Q2	Q3	Q1	Q2	Q3
10	MySQL	0.0019	0.0019	0.0015	25,700	25,764	25,676
100		0.0032	0.0033	0.0035	25,728	25,748	25,792
500		0.0089	0.008	0.0081	25,760	25,760	25,884
1000		0.013	0.017	0.014	25,860	26,100	26,588
10000		0.018	0.13	0.12	25,980	26,748	28,096
10	Neo4j	0.022	0.014	0.017	40,108	40,132	40,120
100		0.062	0.041	0.037	40,276	40,208	40,476
500		0.12	0.11	0.13	40,228	40,344	40,520
1000		0.17	0.26	0.2	40,580	41,100	41,268
10000		0.26	0.85	2.52	40,596	41,432	41,708

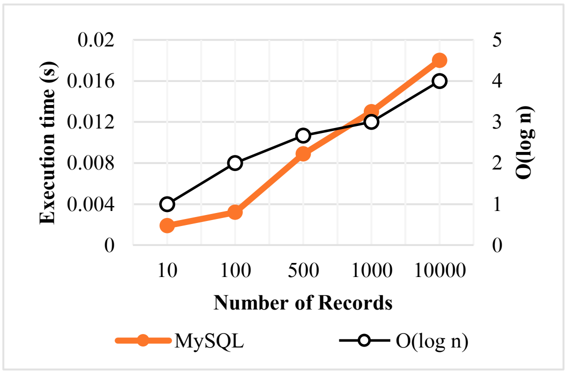


Fig. 6 Time Complexity of Query 1 in MySQL

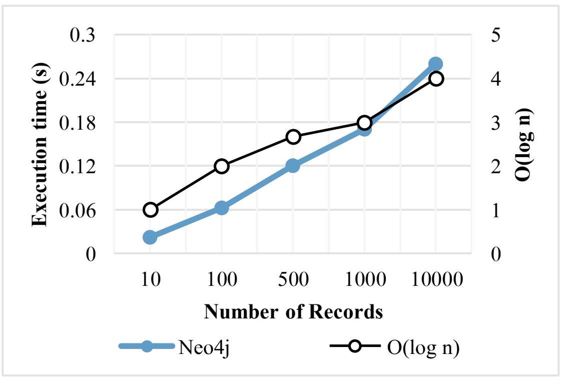


Fig. 7 Time Complexity of Query 1 in Neo4j

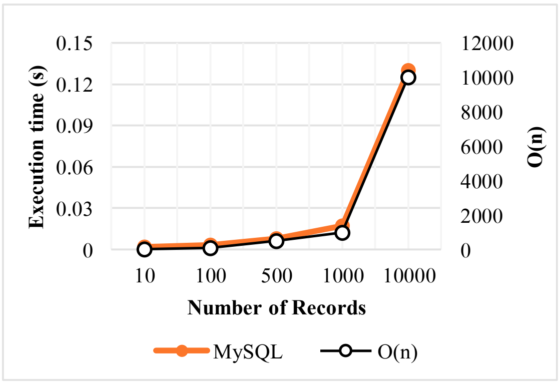


Fig. 8 Time Complexity of Query 2 in MySQL

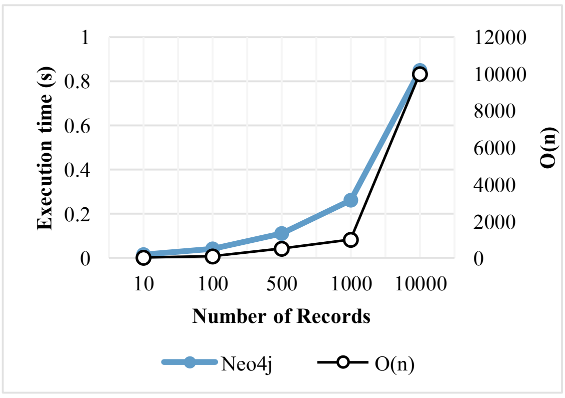


Fig. 9 Time Complexity of Query 2 in Neo4j

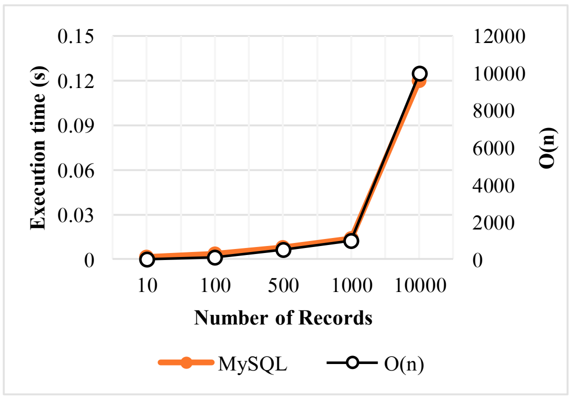


Fig. 10 Time Complexity of Query 3 in MySQL

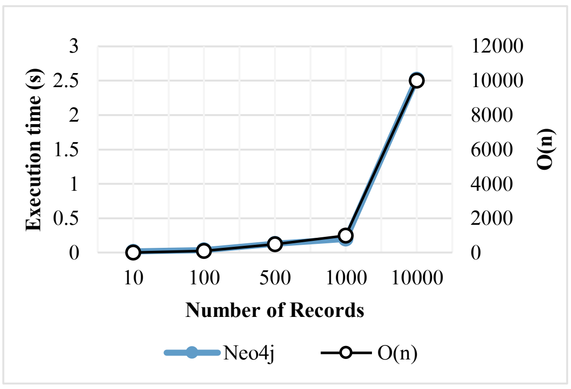


Fig. 11 Time Complexity of Query 3 in Neo4j

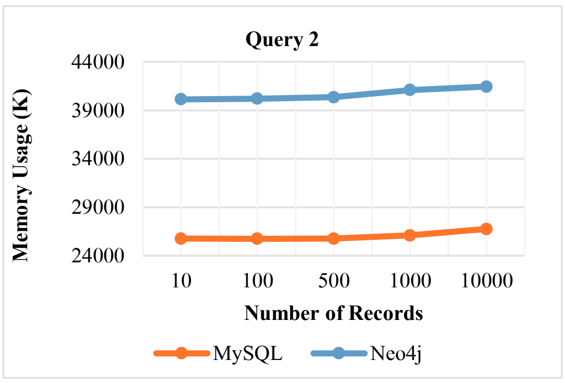


Fig. 13 Memory Usage of Query 2 in MySQL and Neo4j

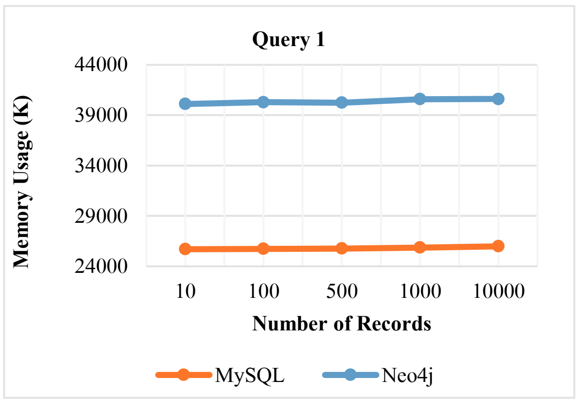


Fig. 12 Memory Usage of Query 1 in MySQL and Neo4j

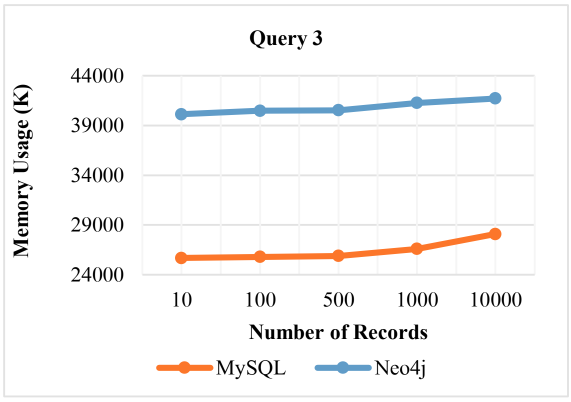
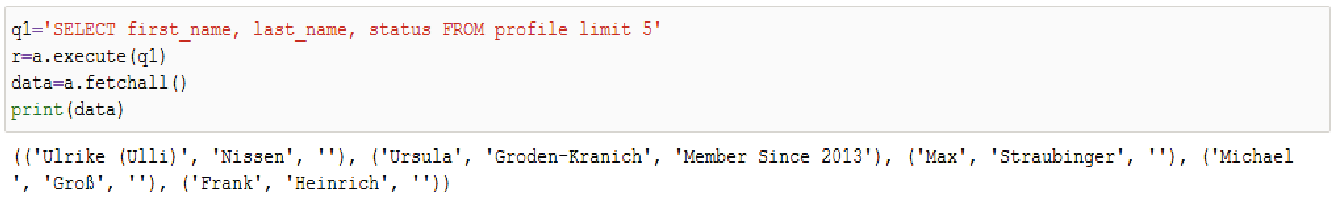


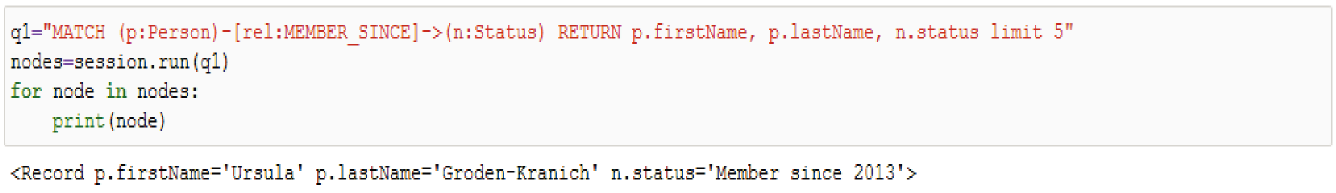
Fig. 14 Memory Usage of Query 3 in MySQL and Neo4j



```
q1='SELECT first_name, last_name, status FROM profile limit 5'
r=a.execute(q1)
data=a.fetchall()
print(data)
```

```
(('Ulrike (Ulli)', 'Nissen', ''), ('Ursula', 'Groden-Kranich', 'Member Since 2013'), ('Max', 'Straubinger', ''), ('Michael
', 'Groß', ''), ('Frank', 'Heinrich', ''))
```

Fig. 15 Execution Result of Query 4 in MySQL



```
q1="MATCH (p:Person)-[rel:MEMBER_SINCE]->(n:Status) RETURN p.firstName, p.lastName, n.status limit 5"
nodes=session.run(q1)
for node in nodes:
    print(node)
```

```
<Record p.firstName='Ursula' p.lastName='Groden-Kranich' n.status='Member since 2013'>
```

Fig. 16 Execution Result of Query 4 in Neo4j

From the table above, it is known that in general the execution time and memory usage in query 1 to query 3, both in Neo4j and MySQL, continue to increase along with the increasing number of records. In this research it is also known that the execution time in MySQL is faster than Neo4j, but both these databases have the same time complexity it is O(log n) for query 1 and O(n) for query 2 to query 3. Also, the memory usage of Neo4j is higher than MySQL, it may happen because MySQL uses data normalization which can eliminate duplication of data. Then from Figs. 15 and Figs. 16, query 4 has shown the ability of databases to handle data with unstructured formats, as we can see in MySQL, query 4 has displayed all Status data in the database, even though there are several fields (Status) whose values are empty. Meanwhile in Neo4j, query 4 only displayed the required data by looking for the relationship between Person who has Status.

V. Conclusion

Both SQL and NoSQL databases have the same function as data storage, but they use different ways which also have different effects for the data being processed, all depending on the needs of the user. This research presents MySQL and Neo4j performance in terms of memory usage and execution time using data public policies decision making by the Deputies. The results of this research indicate:

- In general, the execution time and memory usage increase as the number of records increases.
- MySQL query execution time is faster than Neo4j, however, both these databases have the same time complexity.
- Neo4j has a higher memory usage than MySQL.
- Neo4j has better flexibility than MySQL and suitable for handling data with unstructured formats.

In the future we will try to compare some SQL databases and NoSQL databases with other types using more complex queries and more complex data.

REFERENCES

[1] W. Khan, E. Ahmed and W. Shahzad, "Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases," *International Journal of Advanced Computer Science and Applications (IJACSA) ,* 2017.

[2] H. Leinonen, Simulation analyses and stress testing of payment networks, Bank of Finland payment and settlement system seminars 2007-2008., 2009.

[3] A. Alamsyah and F. Adityawarman, "Hybrid Sentiment and Network Analysis of Social Opinion Polarization," *5th International Conference on Information and Communication Technology (ICoICT),* 2017.

[4] R. Zafar, E. Yafi, M. F. Zuhairi and H. Dao, "Big Data: The NoSQL and RDBMS review," *International Conference on Information and Communication Technology (ICICTM),* 2016.

[5] H. Lu, Z. Hong and M. Shi, "Analysis of Film Data Based on Neo4j," *IEEE/ACIS 16th International Conference on Computer Science (ICIS),* 2017.

[6] A. Oussous, B. F. -Z, L. A.A. and S. Belfkih, "Comparison and classication of nosql databases for big data," in Proceedings of International Conference on Big Data, Cloud and Applications, 2015.

[7] B. Jose and S. Abraham , "Exploring the Merits of NoSQL: A Study Based on MongoDB," *International Conference on Network & Advances in Computational Technologies (NetACT),* 2017.

[8] A. T. Kabakus and R. Kara, "A performance evaluation of in-memory databases," *Journal of King Saud University –Computer and Information Sciences,* 2017.

[9] W. Khan, W. Ahmad, B. Lou and E. Ahmed, "SQL Database with physical database tuning technique and NoSQL graph database comparisons," *IEEE 3rd Information Technology,Networking,Electronic and Automation Control Conference (ITNEC),* 2019.

[10] K. I. Satoto, R. R. Isnanto, R. Kridalukmana and K. T. Martono, "Optimizing MySQL Database System on Information Systems Research , Publications and Community Service," *International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE),* 2016.

[11] "The Neo4j Getting Started Guide v4.0," neo4j, [Online]. Available: https://neo4j.com/docs/getting-started/current/. [Accessed 03 May 2020].

[12] "Cypher Basics I," neo4j, [Online]. Available: https://neo4j.com/developer/cypher-basics-i/. [Accessed 03 May 2020].

[13] [Online]. Available: https://github.com/isnok/neo4j-hackathon. [Accessed 28 October 2019].

[14] [Online] Available: https://social.technet.microsoft.com/Forums/en-US/6608ebac-a7ec-43b0-88d6-e8d95ab8d9cd/memory-usage-vs-memory-private-working-set?forum=perfmon. [Accessed 09 July 2020].

[15] S. Huang, "What is *Big O Notation* Explained: Space and Time Complexity," freeCodeCamp. [Online]. Available: https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt-1674cfa8a23c/. [Accessed 01 August 2020].

[16] S. Debnath, "Analysis of Algorithms | Big-O analysis," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/. [Accessed 01 August 2020].