Analysis of Spatial Database Performance for Location Intelligence

Safiza Suhana Kamal Baharin[1], Patience Obiageli Akunne[2]
[1,2]Centre of Advanced Computing Technology, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia
safiza@utem.edu.my

Abstract— This study aims to examine the performance evaluation of the non-relational spatial database (NoSQL) for location intelligence. NoSQL means Not Only SQL database. It is an unstructured database which is a collection of non-relational data storage systems. Some of the existing NoSQL databases are Cassandra, CouchDB, Hadoop Hbase, MongoDB, etc. Location intelligence as a word is used more often to describe the new generation of GIS. These location-based data require lots of data uploaded into the database in other to keep current and relevant information from source devices (smartphones, laptops, etc.). NoSQL differs from the SQL where SQL stands for Structured Query Language, invented as a standard high-level interface for the management of relational database management systems (RDBMS). Databases based on the relational model include MySQL, MS-SQL Server, Oracle database, etc. The significance of this study is to evaluate NoSQL spatial database performance for location intelligence. The result of the evaluation can help businesses or organizations to know which database will have a better performance in term of location data. Thus, we conduct the experiments to outline the general differences between the SQL and NoSQL and also compare the speed performance of SQL and NoSQL databases for Location Intelligence based on throughput and minimize contention. The latency measured in the experiments shows how long each write or read request takes to be processed. From the result, MongoDB presents better best response time and high-performance throughput (i.e. scalability performance) than Oracle. However, the essential features such as the basic operation of geo-function support that Oracle provides were missing in MongoDB. Relational databases are still far superior if the user needs to calculate geoinformation on the database level. The results provided by this study are significant only for the chosen database settings but indicate that NoSQL databases are a possible alternative, at least for querying information about the attributes. In conclusion, it shows that both SQL and NoSQL databases have advantages and disadvantages when performed spatial queries over location intelligence.

Index Terms—NoSQL, Spatial Database, Location Intelligence

I. Introduction

Databases evolution inception has been began in 1960s, which starting from hierarchical and network databases, object-oriented databases commenced in 1980 and today with SQL and NoSQL databases and cloud databases. Database stores information as a file or a set of files on magnetic disk or tape, optical disk or some other secondary storage device. Information stored in this file can be broken down into records, every single of that contains of one or extra fields. The basic units of data storage are said to be fields, and every single field contains information pertaining to one aspect or attribute of the entity delineated by the database. Using keywords and various sorting commands, users have the option to rapidly search, rearrange, group and select the fields in many records to retrieve or create reports on particular aggregates of data according to different type of databases.

A. Type of Databases

There are six types of databases such as Relational database, Distributed database, Cloud database, Object-oriented database, Graph database and NoSQL database. The first type of database is Relational database. It is a database structured in a tabular form in that data is described so that it can be reorganized and accessed in a number of disparate ways. The structured query language (SQL) is a relational database that is standard for user and application program interface. Second type is Distributed Database. It is stored in several physical locations whereby processing is dispersed or replicated amid disparate points in a network. This kind of database can be homogeneous or heterogeneous. The hardware, operating systems or database applications in a heterogeneous distributed database may be different at each of the locations. Third, Cloud database which provides remunerations such as being able to pay for storage capacity and bandwidth on a per-use basis and it also offers scalability on demand alongside with high availability. Forth, is about object-oriented database. It is a database in that the information or the data to be stored is embodied as an object. Therefore, object-oriented database can be seen as a combination of object-oriented programming (OOP) and database principles. Fifth, is about graph database. This type of database makes use of graph theory to store map and query relationships. Each node denotes an entity and representation of each edge is a connection between nodes. Varieties of applications such social networking applications, recommendation software, bio informatics, content management, security and access control, network and cloud management can be utilized in graph database. The last one is NoSQL database. NoSQL means Not Only SQL database. In other words, it is an unstructured database. This kind of database is effective for big data performance issues in which relational database cannot solve. The fifth and last type of database is classified as non-relational database.

II. Database Performance And Measurement

Database presentation is described as the optimization of resource that is utilized to raise throughput and minimize contention, enabling of the biggest probable workload to be processed. Five factors that impact database presentation are: workload, throughput, resources, optimization, and contention.

A. Workload

It is a combination of online deals, batch jobs, ad hoc queries, and data warehousing scrutiny, utilities, and arrangement commands managed across the DBMS at each given time. Sometimes workload can be predicted (such as heavy month-end processing of payroll, or light access after 6:00 p.m., after which most users should have left for the day), but at other times it is unpredictable. Workload can possess extremely big impact on database performance.

B. Throughput

They overall capability of a computer to process data is known as throughput. Throughput is composite of I/O speed, CPU speed, and parallel capabilities of the machine, and the operating system and system software to be effective.

C. Resources

Software and hardware tools at the disposal of the system. Examples are; memory, disk, cache controllers and microcode.

D. Optimization

Relational database systems are exceptional, such that query optimization is mainly accomplished inner to the DBMS. Some factors that can be optimized such as: SQL formulation, database parameters, and system parameters.

E. Contention

Contention is the condition in that two or extra constituents of the workload are trying to use a solitary resource in a contradictory way. Example, dual updates to the similar piece of data. Higher the contentions lower the throughput.

Database system contributes as one of the enabling forces for company transformations. This arrangement supports enterprise logic and they additionally enable company intelligence. Database performance measure is defined as the process of measuring the performance of a database in real time to know the setbacks and supplementary factors that can cause setbacks in the future. It is a good method to know which database is efficient and has good performance. Five ways to compute database performance employing database statics are as mentioned in [4];

a) Model Statistics: Time model statistics use period to recognize quantitative results concerning specific deeds on the database, such as logon operations and parsing. he most vital period ideal statistic is DB time. DB period is measured cumulatively from the period of instance startup and is computed by aggregating the CPU and pause periods of all sessions not staying on inactive pause events.

b) Active Session History Statistics: Any session that is related to the database and is staying for an event that does not fit in to the idle pause class is believed an active session. Active Session History (ASH) enables you to scrutinize and present methodical scrutiny on both present data in the V$ACTIVE_SESSION_HISTORY view and past data in the DBA_HIST_ACTIVE_SESS_HISTORY view, frequently circumventing the demand to replay the workload to draw supplementary performance information. Object number, file number, and block number pause event identifier and parameters, Session identifier and session serial number, Module and deed term, Client identifier of the session, Service hash identifier and Customer cluster identifier captures data across ASH.

c) Wait Events Statistics: Wait events are statistics that are incremented by a server procedure or thread to indicate that it had to pause for an event to finish before processing might continue. Wait event data reveals different symptoms of setbacks that could be impacting performance, such as latch contention, buffer contention, and I/O contention.

d) Interpreting Database Statistics: When primarily scrutinizing performance data, you can devise possible interpretations of the data by scrutinizing the database statistics. In other to be sure that your interpretation is accurate, go through with other data to institute if a statistic or event is honestly relevant. Because foreground hobbies are tunable, it is suggested to early examine the statistics from foreground hobbies beforehand analyzing the statistics from background activities.

e) Database Time: Database period is described as the sum of the period consumed inside the database processing user request. User's response period is the period interval amid the instant the appeal is dispatched and the instant the response is received. The use of database period will enable one to gauge the performance influence of an entity of the database. Example; [3] shows the Automatic Database Diagnostic Monitor (ADDM) automatically diagnoses the bottlenecks affecting the total database throughput and provides actionable recommendations to alleviate them. ADDM looks at the database time spent in two independent dimensions. The first dimension looks at the database time spent in various phases of processing user requests. This dimension includes categories like 'connecting to the database', 'optimizing SQL statements', and 'executing SQL statements'. The second dimension looks at the database time spent using or waiting for various database resources used in processing user requests. The database resources considered in this dimension include both hardware resources, like CPU and I/O devices, and software resources like database locks and application locks. Fig. 1, shows an example of database time graph.

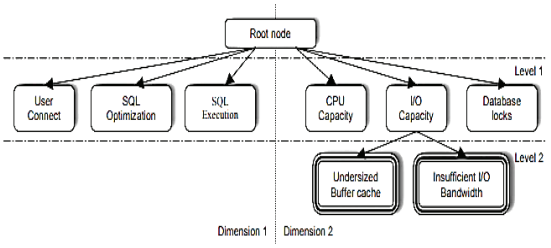

Fig. 1. Sample of DB Time Graph [3]

III. Related Works

The performance of a DBMS plays an integral act in the decision of a firm to use that database. The task needed for performance measurement is a convoluted procedure, countless iterations and adjustments will be demanded to attain the best presentation probable in a given databases. Performance measurement can be utilized to recognize performance bottleneck. A known misinterpretation of performance measurement is that it measures speed. Performance measurement can be grasped out on useful aspects such as ease of use, ease of progress or operational aspects [12]. However, databases with a large number of transactions should focus more on high throughput. The proposed design is derived from a mixture of different needs from the shareholders involved in the use of geospatial data. These shareholders include database users and location intelligence providers.

Table I presents the analysis of related study regarding to Evaluation of NoSQL Spatial Database Performance for

Location Intelligence focuses on performance matrix. Various researches into NoSQL spatial database performance has provided different approaches to evaluate its performance. The comparison of various databases (relational and Non-relational) performance was carried to see which database is good for large amount of data. This paper highlighted database performances while leaving out how NoSQL database can be optimized through the achievement of improving the indices, improvement of the JSON schema and the general database improvement.

TABLE I
RELATED RESEARCH IN SIMILAR TOPICS

Reference	Database and Technology	Metrics & Performance result
[6]	NoSQL database Riak, MongoDB and Cassandra	(Throughput, read and write latency) NoSQL database technology provides benefit of scalability and availability across horizontal scaling, replication and clear data models.
[13]	NoSQL and SQL database. PostrgeSQL and MongoDB.	(Response time, transaction time, execution time and latency) For PostrgeSQL the response time increases with large size of datasets, the response time reaches 200 seconds. For MongoDB the response time is almost similar just differ by a little seconds. MongoDB keeps a high performance even with large datasets whereas; PostrgeSQL performs better with small datasets.
[9]	NoSQL and SQL MongoDB, RDF (Graph database), WFS and MySQL	(Response time, cost, storage capacity and scalability) RDF and WFS, due to large amount of storage data emerging from RDF conversion did not provide a higher performance than initial WFS;
		however MongoDB is seen to be the appropriate solution for sensor data in terms of storage management.
[8]	NoSQL and SQL MongoDB and MySQL	(Total queries per second and average time taken). Relational database with its feature such as schema gave a logical view of the database building which is fast, easy to develop, and eliminated duplication of data but at the same time guaranteed reliability. In NoSQL databases it provided performance and horizontal scalability which made them suitable for data centers requiring massive amounts of storage.
[13]	NoSQL database. HBase and Cassandra, MongoDB and CouchDB, Redis and Voldemort	(Scalability) CouchDB needed a proxy server like RDBMS to achieve scalability; all the other databases were able to provide scalability over multiple nodes. Therefore Scalability was delivered on.
[7]	NoSQL and SQL HBase, MongoDB and Sharded MySQL	(Through-put, CPU IO waits percentage and latency) NoSQL database does not depend only on the configuration of database itself but also depends on the capability of nodes in the database cluster.
[2]	NoSQL and SQL Neo4j and PostgreSQL	(Objective (speed, disk space requirement and scalability. Subjective (maturity/level of support, stability and ease of use)) The measurement included a range of typical spatial queries over the datasets. in some cases, Neo4j should be considered as an alternative for specific tasks.
[11]	DBMS (using 3 schemas). Qosmet Solution	(Execution time (read) Throughput (write)) For throughput, schema 1 and 2 are close to each while schema 3 fell about 33%. For execution time, schema 1 performs best out of the three for some queries. Schema 2 and 3 is consistent than schema 1. Schema 3 has the best overall execution time (read) performance.
[1]	NoSQL and SQL MongoDB and MYSQL	(Read, write and delete) It shows that MongoDB has better performance for most operations eliminating some aggregate functions.
[5]	NoSQL MongoDB, Cassandra and Amazon DynamoDB	(Cost, Response time, Latency, Ease of use, Elasticity and on-demand self-service and Throughput) The NoSQL databases show high performance and scalability.

IV. EXPERIMENT ENVIRONMENT

Databases that can be used to test NoSQL performance with geo-spatial functions include MongoDB and CouchBase. For SQL with geo-spatial function is PostgreSQL and Oracle. The NoSQL performs on the CAP theorem concept (Consistency-Availability-Partition Tolerance). The implementation of the proposed framework will be tested based on scenario and an analysis will be conducted using SQL database (Oracle) in comparison to NoSQL database (MongoDB), which also based on the data requirement of an App-based ride services. This App-based ride services offers ride, lift and on-demand ride service. With this App-based ride service users can indicate their pick-up location and their drop-off location, and sometimes a user can indicate more than one drop-off location. The stakeholders involved in this App-based ride services are Apps user (e.g. customer) and driver. Once a user has indicated his pick-up location and drop-off location, the users expects a fast delivery response from the app by showing that the app found a driver for the user and in some cases, it might be difficult to find a driver. When the driver is found, being able to use the map to find the exact location of the user. Fig 2 shows the entity relationship diagram (ERD) contains entity, attributes, constraint, datatypes and data size which will be stored in the Oracle database as tables, rows and columns.

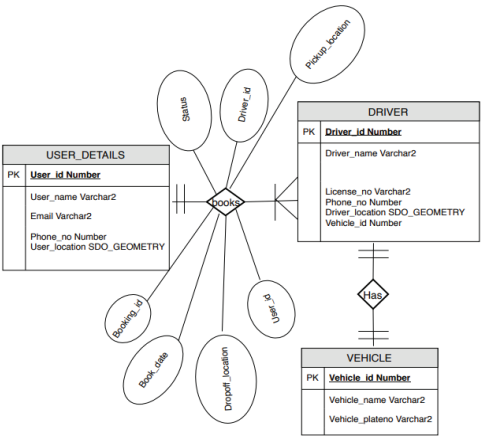

Fig. 2. ERD for Test Scenario for Relational Database

Fig. 3 shows how the data will be stored in MongoDB used JSON. In order to evaluate the two databases, we used a system that has an Intel (R) Pentium (R) CPU N3540 @ 2.16GHz with 4GB hard disk and windows 7 Ultimate 64bit operating system to conduct our experiment. We came up with a test scenario which we used to design our data structure for both databases. Oracle 12c Express Edition is installed for SQL database and MongoDB 3.6.4 2008R2Plus Enterprise for NoSQL database.

In other to carry out our testing we wrote three (3) complex SQL queries in term of distance to test response time of the both database in five scales of 5000, 20000, 50000, 75000 and 100000 records. For throughput, the metric was tested using `SELECT` and `DELETE` for both database in five scales of 5,000, 20,000, 50,000, 75,000 and 100,000 records. For Oracle Database PL/SQL is written to automatically populate data into the database and for MongoDB we generated JSON data online using `InsertMany`. We inserted 2000 data each. In this paper, we measure the write and read latency, as well as the throughput.

```
User_details

{
 "type": "object",
 "definitions": {},
 "$schema": "http://json-schema.org/draft-07/schema#",
 "properties": {
  "_id": {
   "$id": "/properties/_id",
   "type": "string",
   "title": "The _id Schema ",
   "default": "",
   "examples": [
    "5b00dc09a1d7c7d4a6a4ea83"
   ]
  },
  "user_id": {
```

Fig. 3. NoSQL Data Schema for Test Scenario

The latency measured in these experiments shows how long each individual write or read request takes to be processed. It does not include network latency between the load generator and the database cluster. Instead, it is measured from the database perspective, i.e., the time that is required to process a single request and result output. Set of queries are written to test the response time/latency and the throughput is measured based on CRUD and the seconds it takes for the request to be processed and also the response time/ latency and throughput is measured in scale of number of records inserted and maximum of five (5) seconds. Table II shows the details of different features of the database management systems that we are evaluating.

The SQL query used to evaluate the spatial performance of SQL (in Oracle) and NoSQL (MongoDB) database are based on these three (3) queries:

- **Q1**:*To find driver where distance is 10, 15, 25, 30, 35, 40, 45 and 50 kilometers*.
- **Q2**:*To find total Number of Trips made by Driver*.
- **Q3**:*To find total Number of Trip where Status is completed or Not*.

TABLE II
DATABASE FEATURE

	Oracle Database	Mongo DB
Spatial query	Oracle Lucene index	Spatial query
Extension module to support distributed computing	Natively distributed	Natively distributed
Language	Java	C++
Query Language	SQL	Json
Data-type	SDO_GEOMETRY	GeoJSON
Partitioning Method	Sharding	Sharding
Transaction Property	ACIDE/BASE	BASE
License	Apache 2.0	GNU AGPL v3.0
Version	12.2.0.1	3.6.4
Data Model	Column +row relational DBMS	Document +key
First Release	2016/2017	2018

The SQL statement used to find the distance of the driver in kilometers using the spatial function in Oracle database and MongoDB to find the drivers distance is shown in Fig. 4 and Fig. 5 respectively.

```
SELECT d.driver_name, d.phone_no,
sdo_geom.sdo_distance (d.driver_location,
u.user_location, 1, 'unit=km') as distance
FROM driver d, user_details u
WHERE u.user_id=5
AND sdo_within_distance(d.driver_location,
u.user_location, 'distance=10 unit=km')=
'TRUE';
```

Fig. 4. SQL Statement to Query Distance in Kilometer for Q1

```
db.Drivers.find({driver_location: {
  $near:
{ $geometry:
        {
        type:"Point",
       coordinates:[-69.281779, -61.805321]
    },$maxDistance:40000
}
    }
  })
```

Fig. 5. Query Statement for NoSQL for Q1

The SQL statement to Q2 in Oracle database to find the total number of trips made by the driver by using the aggregate function `sum` and `count` to find the total trips in the *Booking* table is shown in Fig. 6.

```
SELECT count (b.booking_id) as total,
d.driver_name as name, sum
(sdo_geom.sdo_distance
(b.pickup_location,b.dropoff_location,1,'unit=
km')) as covered
FROM driver d join booking b on
b.driver_id=d.driver_id
WHERE d.driver_id=390002
GROUP BY d.driver_name
ORDER BY total;
```

Fig. 6. SQL Statement to Query Distance in Kilometer for Q2

The SQL statement to Q3 in Oracle database to find a total trip made by driver if completed or not is shown in Fig. 7.

```
SELECT count(booking_id) as total, status as
status, sum (sdo_geom.sdo_distance
(pickup_location, dropoff_location, 1,
'unit=km')) as covered
FROM booking WHERE status=0 OR status=1 OR
status=2 GROUP BY status ORDER BY total;
```

Fig. 7. SQL Statement to Query Status is Completed or Not for Q3

To evaluate the database performance two metrics are used; throughput which is the average number of operations per seconds and latency which is average time required to perform a single operation. Both, throughput and latency equation are shown in (1) and (2) respectively.

$$Throughput = \frac{(number\ of\ operations\ per\ thread)*(number\ of\ threads)}{average\ execution\ time\ per\ 1\ thread} \quad (1)$$

$$Latency = \frac{average\ execution\ time\ per\ 1\ thread}{number\ of\ operations\ per\ thread} \quad (2)$$

V. Result Analysis And Discussion

The vast amount of stored data is seen as a characteristics of today's high-tech. Spatial data fulfill the criteria of fast changing, colossal datasets that in the end makes constant indexing of data necessary. It is vital to understand the presentation if NoSQL and SQL can grasp this rising increase of colossal datasets.

A. Result on Response/ Latency

Table III and IV shows the distance and the scale of the record in 5000, 20,000, 50,000, 75,000 and 100,000 in Oracle and MongoDB for Q1. The SQL query in Fig. 4 and Fig. 5 is run to find distance of the driver for 10, 15, 25, 35, 40, 45 and 50 kilometers and the time (in second) is recorded.

TABLE III
Distance And Scale Of The Record For Number of Record 5000 And 2000

Distance	Number of Data			
	5000		20000	
	Oracle	MongoDB	Oracle	MongoDB
10	0.01	0.02	0.02	0.02
15	0.01	0.03	0.04	0.03
25	0.03	0.03	0.08	0.03
30	0.03	0.03	0.13	0.03
35	0.04	0.03	0.17	0.03
40	0.05	0.03	0.22	0.03
45	0.07	0.03	0.26	0.03
50	0.09	0.03	0.32	0.03

TABLE IV
Distance And Scale Of The Record For Number of Record 50000, 75000 And 2000

Dist	Number of Data					
	50000		75000		100000	
	Oracle	Mongo DB	Oracle	Mongo DB	Oracle	Mongo DB
10	0.30	0.02	0.32	0.03	0.38	0.04
15	0.34	0.03	0.34	0.03	0.41	0.04
25	0.36	0.03	0.34	0.03	0.43	0.05
30	0.36	0.03	0.38	0.04	0.46	0.05
35	0.40	0.03	0.39	0.04	0.47	0.05
40	0.42	0.03	0.40	0.04	0.48	0.05
45	0.41	0.03	0.43	0.04	0.50	0.05
50	0.46	0.03	0.48	0.04	0.51	0.06

The results show that there is better performance for NoSQL as data size increases, whereas SQL fails at a very colossal dataset. SQL increases exponentially as size of dataset increases, whereas NoSQL still performs within some bound. Index querying for NoSQL and SQL differs. NoSQL uses a 2D-sphere index and SQL uses GST (Generalized Search Tree) index. Based on query for attribute information, the performance for NoSQL response time is less than the response time for SQL databases, because the SQL databases will need more time to answer queries. In term of queries using geo-function `within`, the datasets for NoSQL doesn't play much of a big role. The response time for SQL and NoSQL is almost linear; NoSQL differs by some seconds [10].

Table V shows the result of the time recorded (in second) from the Q2. While the distance between two geographical points can be computed using the "`sdo_distance function`" in contrast using native functionality MongoDB does not offer any functionality to calculate this distance. The solution to this inability is selecting the data and manipulating the calculation on the client-side server.

TABLE V
Number Of Trips

NO.OF RECORDS	TIME (SECONDS)
5000	0.01
20000	0.03
50000	0.06
75000	0.07
100000	0.07

Table VI shows the result of the time recorded (in second) from the Q3.

TABLE VI
TOTAL TRIPS

NO.OF RECORDS	TIME (SECONDS)
5000	9.67
20000	37.47
50000	93.46
75000	97.98
100000	106.86

B. Result on Throughput

The results of the throughput evaluation for both Oracle and MongoDB are shown in Table V to Table VII. In this study, the throughput was conducted for `SELECT` and `DELETE` statements to test the response time using a scale of data records in 5000, 20,000, 50,000, 750,000, and 100,000. A Driver, User, and Booking was an entity from the SQL data structure in Figure 2 and NoSQL data schema in Figure 3.

TABLE V
ORACLE AND MONGODB THROUGHPUT FOR DRIVER

DATA	ORACLE		MONGODB	
	SELECT	DELETE	SELECT	DELETE
5000	1.47	8.40	0.01	1.81
20000	0.01	46.99	0.01	1.92
50000	0.01	185.13	0.01	5.48
75000	0.01	223.12	0.01	6.56
100000	0.01	474.48	0.02	7.12

TABLE VI
ORACLE AND MONGODB THROUGHPUT FOR USER

DATA	ORACLE		MONGODB	
	SELECT	DELETE	SELECT	DELETE
5000	1.27	8.43	0.01	1.79
20000	0.01	47.82	0.01	1.94
50000	0.01	194.89	0.01	3.67
75000	0.01	223.17	0.01	5.43
100000	0.01	483.84	0.02	6.24

TABLE VII
ORACLE AND MONGODB THROUGHPUT FOR BOOKING

DATA	ORACLE		MONGODB	
	SELECT	DELETE	SELECT	DELETE
5000	1.72	0.24	0.01	2.01
20000	0.01	0.88	0.01	2.01
50000	0.01	2.13	0.01	4.68
75000	0.01	2.66	0.01	7.56
100000	0.01	9.20	0.02	7.78

To get time in the seconds, we ran each query five times and then divided the result by five to get the average. The experimental findings are plotted based on time (seconds). Fig. 8(a) to Fig. 8(h) showing the response/ latency time for scale 10km, 15km, 25km, 30km, 35km, 40km, 45km, and 50km respectively.

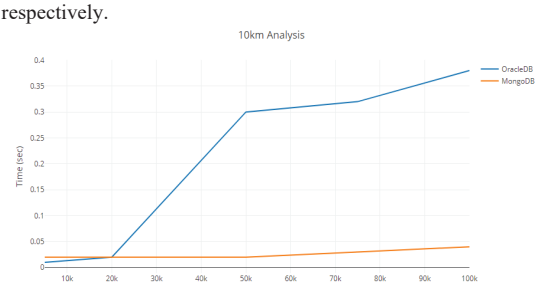

Fig. 8 (a) the response/ latency time for 10km

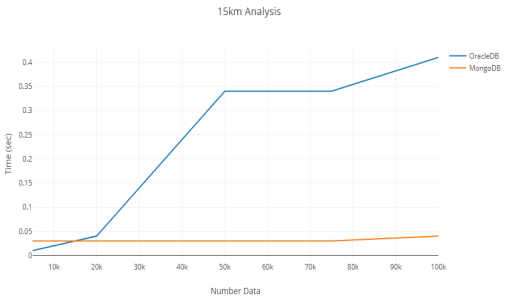

Fig. 8 (b) the response/ latency time for 15km

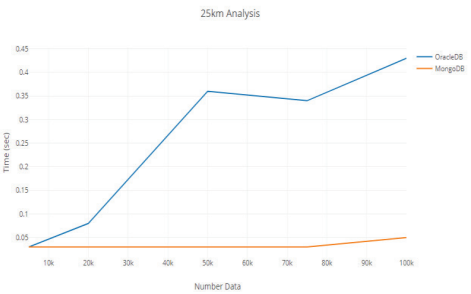

Fig. 8 (c) the response/ latency time for 25km

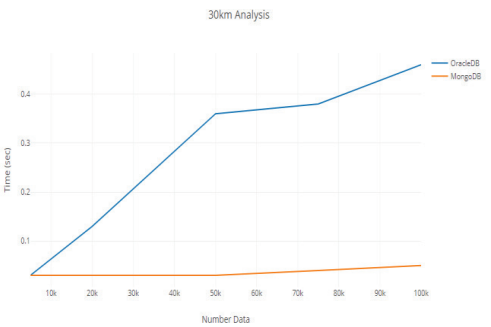

Fig. 8 (d) the response/ latency time for 30km

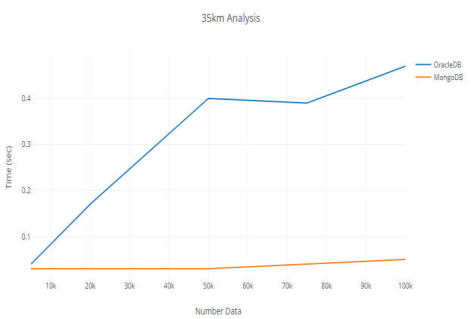

Fig. 8 (e) the response/ latency time for 35km

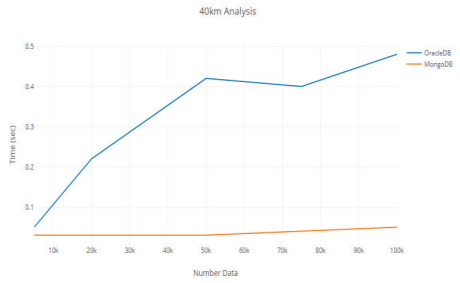

Fig. 8 (f) the response/ latency time for 40km

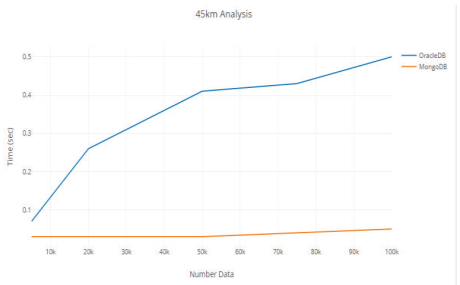

Fig. 8 (g) the response/ latency time for 45km

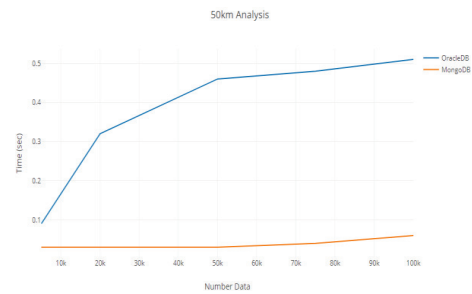

Fig. 8 (h) the response/ latency time for 50km

Fig. 8(a) shows that MongoDB has a faster response time than the Oracle database. At first, the Oracle had fast response time for 5000 to 20,000 records, however, from 20,000 records and above MongoDB gives a steady result. Fig. 8(b) to Fig. 8(d), shows MongoDB outperforms Oracle Database. Oracle has fast response time for smaller data from the scale of 5000 to at most 10,000 records. When it comes to applying index on the spatial data type in Oracle with records above 50,000 records, the tablespace in Oracle cannot contain more records due to the limitation of Oracle Express Edition. Generally, MongoDB has a fast response time for large records and shows better performance for spatial data. Fig. 8 (e) to Fig. 8(f) shows that MongoDB has a better faster response time than Oracle Database. Oracle database has a fast response time when it has to do with querying a single row. For Fig. 8 (g) to Fig. 8 (h), it shows that as the distance to calculate increases, the response time for the Oracle database is delayed. Oracle database takes a long time to output results, whereas, MongoDB no matter the distance, has a faster response time. Fig. 9 shows the results for the `SELECT` statement for User Analysis in the Oracle database. It shows that for 5,000 records, the Oracle provided a low throughput performance, but it shows a high throughput performance from 6,000 to 100,000 records. In the meantime, it shows high throughput performance for the five scales for MongoDB. MongoDB outperforms the Oracle database for the `DELETE` statement.

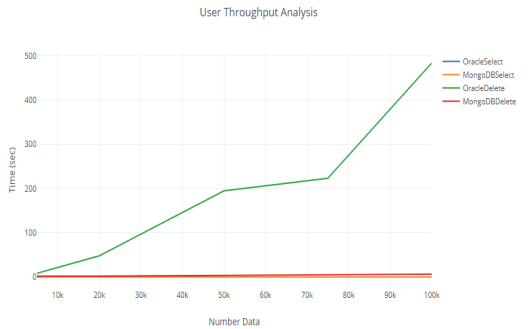

Fig. 9 User Analysis

Fig. 10 shows the result of the Oracle `SELECT` statement for Driver Analysis. It shows low throughput performance for 5,000 records but shows high throughput from 6,000 to 100,000 records. MongoDB maintains a high throughput from the first to the fifth scale. MongoDB has a higher throughput performance than the Oracle database for the `DELETE` statement.

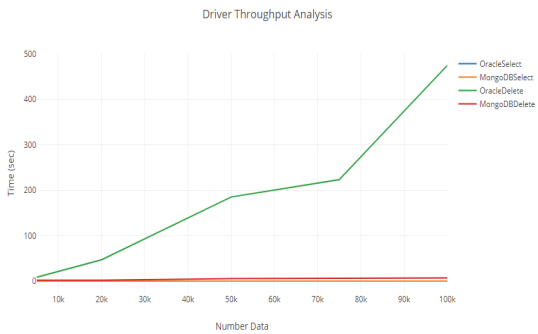

Fig. 10 Driver Analysis

The result of the `SELECT` statement for Booking Analysis in Oracle is shown in Fig. 11. The result shows low throughput performance for 5,000 records but maintains a high throughput from 6,000 to 10,0000 records. MongoDB has a high throughput performance for `SELECT` statements. For the `DELETE` statement, the Oracle outperforms MongoDB but, generally shows that Oracle takes a longer time to respond to the `DELETE` statement. Therefore, MongoDB has a high throughput performance rather than Oracle.

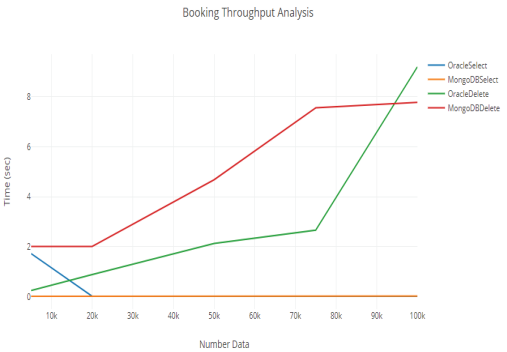

Fig. 11 Booking Analysis

VI. Conclusion

In conclusion, MongoDB outperforms Oracle in terms of scalability, allowing additions and improvements without any essential changes to the database. In the meantime, this will have a significant impact on Oracle's database. Since Oracle is a relational database, the scalability of this database can be a bit difficult. It is because some tables are linked to each other in terms of the parent-child relationship. Data cannot be added without first referring to the parent key (primary key) and before attending the child key (foreign key), but, MongoDB, which is non-relative, it has a better scalability performance. The Oracle database does not have the best response time and high-performance throughput. Both databases Oracle and MongoDB have similar speeds for the `SELECT` statement. During the experiment, to query about the distance, the index was created for columns with a spatial data type. However, while conducting the `DELETE` statement, the index was not dropped first. Hence, it resulted in the `DELETE` statement taking a too long run. Due to this process, MongoDB has a higher throughput than Oracle. However, MongoDB has its weakness when aggregate functions are done on non-key attributes. It has a limitation function of geo-proximity search queries (which does not natively exist) compared to the Oracle database. Thus, it makes the Oracle database still far superior if the user needs to calculate geoinformation on the database level especially, when it's related to the location intelligence.

Acknowledgment

The authors would like to thank CIT, Center for Advanced Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka for giving opportunity and tools to conduct this study.

References

[1] Aboutorabi, Seyyed & Rezapour, Mehdi & Moradi, Milad & Ghadiri, Nasser. (2015). Performance evaluation of SQL and MongoDB databases for big e-commerce data.
[2] Bart Baas (2012). Geographical Information Management and Applications [online] Available at: http://www.gdmc.nl/publications/2012/Neo4j_versus_PostGIS.pdf [Accessed 18 Apr. 2018].
[3] Dias. K, Ramacher. M, Shaft. U, Venkataramani. V, Wood. G. (2005) Automatic Performance Diagnosis and Tuning in Oracle. Proceedings of the CIDR Conference. Docendo. p. 196.
[4] Docs.oracle.com. (2018). Measuring Database Performance. [online] Available at: https://docs.oracle.com/database/121/TGDBA/measure_db.htm#TGDBA146 [Accessed 1 Jun. 2018].
[5] George Antoniou, Evaluation of Major NoSQL Databases, Master Thesis, The University of Edinburgh, 2016.
[6] Gorton, Ian & Klein, John. (2015). Performance Evaluation of NoSQL Databases: A Case Study.
[7] Harish Balasubramanian. (2014). Performance Analysis of Scalable SQL and NOSQL Databases: A Quantitative Approach.
[8] Hadjigeorgiou, C. (2013). RDBMS vs NoSQL: Performance and Scaling Comparison.
[9] Mehfooz, Muhammad Arslan. SenseMark – A database benchmark and evaluation study for alternative databases for Sensor data and IoT. Master thesis, University of Oslo, 2016
[10] Reinhardt, Wolfgang & Schmid, Stephan & Gálicz, Eszter. (2015). Performance investigation of selected SQL and NoSQL databases.
[11] Ruottinen, P. (2015). Database architecture design and performance evaluation for Quality of Service measurements: Master's thesis. University of Oulu.
[12] Sawyer, T. (1992). Doing your own Benchmark, in Benchmark Handbook: For Database and Transaction Processing Systems, edited by J Gray, San Francisco: Morgan Kaufmann Publishers Inc.
[13] Stephan Gerhard Schmid, Eszter Galicz, Wolfgang Reinhardt (2015), WMS performance of selected SQL and NoSQL databases, Computer ScienceInternational Conference on Military Technologies (ICMT)
[14] Toit, P.D. (2016). An evaluation of non-relational database management systems as suitable storage for user generated text-based content in a distributed environment.

Safiza Suhana Kamal Baharin, raduated with Bachelor of Science in (Geoinformatics) and Master of Science in Geoinformatics from University of Technology of Malaysia (UTM), Skudai. She received her PhD in Information and Communication Technology from Universiti Teknikal Malaysia Melaka (UTeM). As senior lecturer in Software Engineering Department, she explores various research themes related to Database, software engineering and ICT, specifically in Spatial Databases, Geoinformatics, and Spatio-temporal Analysis.

Patience Obiageli Akunne, graduated with Master of Computer Science (Database Technology) in the Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM).