International Journal of Multidisciplinary and Current Research ISSN: 2321-3124
Research Article Available at: http://ijmcr.com

Comparative Analysis of Relational (Oracle) and Non-Relational (Cassandra) Databases for Business Intelligence

Toluwalope Mary Akinmoladun[1*], Peter Lake[1], Oluwarotimi Williams Samuel[2] and Konstantinos Domdouzis[1]

[1]Department of Computing, Sheffield Hallam University, Sheffield, UK, England
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China

Accepted 22 March 2017, Available online 23 March 2017, **Vol.5 (March/April 2017 issue)**

Abstract

The need for business intelligence systems (BI) cannot be overemphasised because of the huge data constantly being generated in the daily operations of business organisations and the opportunity provided to discover new insights for the improvement of organisational effectiveness and efficiency from the data. This study attempts to carry out performance related tests on Oracle and Cassandra in order to propose a suitable database for business intelligence. Firstly, the extract, transform and load (ETL) processes was used to move data into Oracle and Cassandra virtual machines. Secondly, SQL and NoSQL queries were run on the data in three iterations to test for performance in selected workloads (Create and load process, read, update, delete and join operations) both before and after query optimisation. To create a common ground for comparison, similar queries were run on similar datasets on both databases. Then the results from the tests were statistically analysed using Microsoft Excel. Experimental results show that the latency values of Oracle are observed to be lower than that of Cassandra, accuracy values of Cassandra are observed to be nearly the same with that of Oracle in the create and load process, while their accuracy values are observed to be slightly different in the remaining tested workload, and the throughput values of Cassandra are observed to be higher than that of Oracle. Also, the extent to which these performance outcomes support data analytics for BI is hereby presented.

Keywords: *SQL, NoSQL, Cassandra, Oracle, Business Intelligence and CQL.*

1. Introduction

The emergence of standards in computing, automation, and technology involved in modern businesses have led to the geometric generation of vast amounts of electronic data. Business organisations now depend on such huge amount of data to provide feedback and foundational information about their operational environment. According to Connolly and Begg [1], Business intelligence (BI) encompasses the processes for collecting and analysing data, the technology used in collecting and analysing the data, and the information determined from these processes with the intention of facilitating corporate decision making. In other words, BI brings together a wide range of analytical software and solution for collecting, analysing, and providing access to relevant information needed to make better managerial decisions [2]. The term business intelligence was first used by Hans Peter Luhn in 1958 in an article published for IBM, which he defined as the ability to perceive the existence of interrelationships in presented facts in such a way that enhances proactivity and guides action towards a desired goal. For more than 20 years, the term was not used until Howard Dresner who was an analyst with Gartner re-introduced it in 1989. Dresner's definition of BI is similar to how it is being used today and since then, it has become widely accepted. The term decision support is also being used interchangeably with BI although not popularly used as the latter, it conveys a better literal meaning [3].

The reality of social and economic factors faced by contemporary sectors of industries has made organisations to seek for equipment that would enhance effective acquisition, processing and analysing of large amounts of data that are being generated from heterogeneous sources and that will make it possible to identify and predict patterns and trends which will serve as the basis for discovering new knowledge. Knowledge workers in organisations need to make decisions under time pressure, monitor competition and possess different views about their organisational information, carry out complex analysis of data from different sources and consider the different variants needed for their organisation's performance. Therefore, organisations need Business Intelligence (BI) systems to perform these tasks effectively and efficiently. More of its tasks includes intelligent exploration, integration, aggregation and multidimensional data analysis [4].

Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijmcr/v.5.2.6

Business intelligence (BI) software comprises of decision support technologies intended to enable knowledge workers such as analysts, managers and executives to make useful and effective decisions faster. Several industries have adopted the BI software technology in the past two decades as a result of the performance both in the number of products and services derived from effectively harnessing the BI software. However, the need for the BI software has majorly increased because enterprises no longer acquire and store very large amounts of data from different sources alone, they now collect data at a finer granularity to meet their specific needs. This in turn generate a larger volume of data [5]. It is difficult to find a successful organisation that has not leveraged the BI software technology for its business data. It has been applied to different sectors such as manufacturing, retail, financial services, transportation, telecommunications, utilities and in health care.

Intelligent data analysis techniques are now deployed to enhance faster business decisions and deliver customised functionalities to customers [5]. Zeng *et al*. [6] and Nedelcu [7] proposed that accurate, valid, integrated and timely data are needed for the successful application of BI in a business enterprise. Elbashir [8] reported that because BI systems belong to an important class of data analysis and reporting which provides decision makers with timely and relevant information, organisations need to integrate their BI systems into management and operational processes. Chaudri *et al*. [5] further asserted that as a result of the different sources of data and its inconsistent format, problems of data integration, cleansing and standardisation are posed for pre-processing the data required for BI tasks. For example, data stored in Oracle and Cassandra databases are of different formats and structure. Thus, efficient and scalable data loading capabilities are highly imperative to efficiently extract and integrate (create and load, read, update and delete operations) the data into BI systems.

Moniruzzaman and Hossain [9] noted that the digital world is growing very fast and becoming more complex as data generation has increased in volumes (terabyte to petabyte), variety (structured, unstructured and hybrid), and velocity (high speed in growth). This is a global phenomenon commonly referred to as Big Data, and such data cannot be effectively managed for BI by using the conventional data management tools such as the relational data management systems. In order to address this problem, a number of Structure Query Language (SQL) and Non Structure Query Language (NoSQL) tools have been proposed as alternative means of analysing such big data. SQL databases such as Oracle operate on fixed table structures in which data can only be selected/retrieved using only the SQL. These kind of databases often employ one or more join operations to select/retrieve data across multiple tables. In addition, SQL driven databases scale well in a vertical manner and worse in a horizontal manner. On the other hand, NoSQL databases such as Cassandra typically require key-value stores, which allows data to be stored and retrieved by key. Because it does not support the fixed data structure, less powerful query languages are required to retrieve the stored data. In contrast to SQL databases, the NoSQL databases scale well along the horizontally [10]. Despite the huge advances made in the field of big data analytics, only few studies had focused on evaluating the performance of NoSQL and SQL tools with respect to big data analyses for BI. Hence, deciding the most appropriate database solution for BI tasks such as querying, storage, and security [11], is still a major challenge.

In this research, the performances of SQL and NoSQL tools were systematically investigated and compared with respect to (1) accuracy, (2) average latency, and (3) throughput, for data analytics by using structured open data developed for a BI system. The possibilities of using relational (SQL) and non-relational (NoSQL) databases for data management tasks were also discussed. As underlying databases for the developed BI system, Oracle is considered as a representative of SQL database because it enables its users perform administrative functions such as creating schema objects including tables, views and indexes, granting privileges to users, managing users' security, managing database memory and storage, importing and exporting data, viewing performance and status information of the database. In addition, it provides a platform for database performance tests with the help of utilities like SQL*Loader [12]. On the other hand, Cassandra was chosen as a representative of NoSQL database because it is a scalable open source NoSQL database that is developed for managing large amounts of structured, semi-structured, and unstructured data across multiple data centres and the cloud. Moreover, Cassandra architecture allows authorised users to access data and connects to any node in the data centre using the Cassandra Query Language (CQL) which is similar to the SQL. It also provides a platform for importing and exporting data using the Cassandra utility (cqlsh) and viewing performance reports about a database [13].

The rest of this paper is organised as follows: Section 2 presents a critical review of existing related research on SQL and NoSQL databases for BI. Section 3 highlights the systems specification, details of the software used in the experimental framework and the steps taken to execute the performance tests. Section 4 presents and discusses the obtained and Section 5 concludes the paper presents future research direction.

2. Literature Review

Lee *et al*. [14] conducted a research to investigate and evaluate the suitability of NoSQL and the document-centric data structure of Extensible Mark-up Language (XML) for structured clinical data and revealed that in terms of query speed, NoSQL performed better than XML, although they both demonstrated the potential of

becoming key databases for clinical data management. They concluded that implementing NoSQL approach on a relational database offers database developers the opportunity of using a schema-less and non-relational design for handling complex data while maintaining existing well-established relational database systems. NoSQL database however, falls short of scalability and flexibility when compared to XML approaches in their study.

Veen *et al*., [10] compared the relative performance of an SQL database (PostgreSQL) and two NoSQL databases (Cassandra and MongoDB) with regards to sensor data storage and their obtained results show that Cassandra is best suited for large critical sensor applications because it is built to scale horizontally, MongoDB is best suited for small or medium sized non-critical sensor application, particularly when write performance is important, meanwhile PostgreSQL is best suited when flexible query capabilities are required and read performance is important.

Hecht and Jablonski [15] investigated the underlying techniques of NoSQL databases in relation to their applicability for certain data analytics requirements in BI by comparing their data models, query possibilities, concurrency controls, partitioning, and replication opportunities. They eventually recommended that key value stores should be used when fast and simple operations are needed, document stores should be used when a flexible data model with great query possibilities is important, column family stores should be used for large datasets requiring scaling at a large size and graph databases should be used in domains where data entities are as important as the relationship that exists between them.

Grolinger *et al*., [11] conducted a review on NoSQL databases and SQL databases because of the growing amounts of large data generated daily resulting in increased data processing and the varying BI tasks required. He pointed that in terms of querying, SQL pattern of querying has been adopted in the NoSQL world because of its widespread usage over the past years. For example, Cassandra offers a similar variant such as Cassandra Query Language (CQL). In terms of scaling, Cassandra has been recognised to be capable in handling large number of write requests. In terms of security, NoSQL solutions have been affirmed not to be as mature as those in traditional relational database systems. When Cassandra was compared to MySQL, it achieved the highest throughput in update operations on heavy workload. It also achieves a good throughput of 50% on read-write workloads and 99% on write workloads [16].
Tudorica and Bucur [17] pointed that although, NoSQL databases were created to offer a higher performance in speed and size and a higher availability at the price of replacing the ACID (Atomic, Consistent, Isolated, Durable) trait of relational databases with a weaker BASE (Basic, Availability, Soft state, Eventual Consistency) trait, it is suggested that they cannot be used interchangeable but should rather be chosen depending on the problem at a given instance.

Li and Sathiamoorthy [18] conducted a study to investigate the performance of some NoSQL and SQL databases by comparing read, write, delete and instantiate operations on key-value stores and discovered that not all NoSQL databases perform better than SQL databases as some are much worse. For each database, performance varies with each operation and there is a relationship between performance and the data model each database uses. For instance, Cassandra is slow on read operations but reasonably good on write and delete operations. This implies that data retrieval will be slow while data creation and updating as well as deletion are reasonable faster when data analytics for BI is performed in Cassandra database.

However, a comparative analysis of the performances of Oracle (a representative of SQL database) and Cassandra (a commonly used NoSQL database) have been rarely investigated especially for BI tasks. Therefore, this study is aimed at investigating the performance outcomes in terms of accuracy, average latency, and throughput of Oracle and Cassandra for structured data management using similar datasets and workloads on create and load process, read, delete, and update operations. The results of this study may provide proper insight on the most suitable database for data analytics in business intelligence.

3. Performance Comparison of Relational (Oracle) and Non-Relational (Cassandra) Databases

In order to carry out the experiments on Oracle and Cassandra databases, both test environments are implemented using similar experimental setups. The setups make use of two (2) separate Virtual Machine (VM) Workstation 11 hypervisor which subsequently hosts one Ubuntu VM operating systems (Oss) each. Oracle database software application was installed on the first Ubuntu VM while the Cassandra database software application was installed on the second Ubuntu VM. The performance metrics used in the current study include Average Latency Test (ALT), Accuracy Test (ACT) and Throughput test (THT).

3.1 System specifications for the experimental setups

Both Cassandra and Oracle VM's were installed on the same Workstation, therefore the system specification remains the same as shown in Table 1 and 2.

Table 1 System specifications

S/N	System	Configurations
1	System Model	HP Z210 CMT Workstation
2	Processor	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
3	Installed-Memory	16.0 GB
4	System type	64-bit Operating System
5	Hard Drive	1 TB
6	Operating System	Window 7 Enterprise
7	Hypervisor Application	VMware Workstation 11.1.2

Table 2 VM specifications

S/N	Virtual machine	Configurations
1	Hard Disk size	20.1 GB
2	Internal Memory	1 GB
3	Processor	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
4	System type	32-bit Operating System
5	Operating System (Oracle)	Ubuntu 10.04 LTS
6	Operating System (Cassandra)	Ubuntu 11.10 LTS

3.2 Performance test execution on Oracle database virtual machine

A number of SQL queries were run against small and large datasets and migrated using the extract, transform, and load processes (ETL) to test for performance in create and load process, read, update, and delete operations before and after optimisation. All the queries were run in three iterations and the average was computed. The processes involved in the individual performance tests are described as follows.

3.2.1 Average latency tests (ALT)

The processes for the performance tests execution on small and large datasets before optimisation are quite similar. For the create process, a Data Definition Language (DDL) script was created and saved in the test directory. On the VM terminal, the *SET TIMING ON* was typed and at the prompt the DDL script was run to create tables for both small and large datasets. For the load process, a control file was created and saved to load source data in *.csv* format into the already created tables. This file was thereafter run in the VM terminal with the timing and records of the load process stored in the corresponding log files. For the read, update, delete and multiple join operations, an SQL script was created to select, update, and delete data respectively across the created database, then saved in the performance test directory. Thereafter, a timing template script for each operation was created to store the latency results (time required to query the database system and get the desired results or the delay and waiting time experienced by a user during query request) and saved in the test directory. This script was then run on the VM terminal and the timing results as well as the retrieved data were stored in the corresponding log files.

For the performance tests execution on the small and large datasets after optimisation, an index was created for the columns that are in the where clause for select and join SQL statements and saved in the performance test directory. Similarly, the same SQL queries used for the read, update, delete and multiple join operations on the small and large datasets before indexing were also run on the datasets after indexing. The timing template scripts for each operation was also created and run on the VM terminal to store the latency results of retrieved data from the database as presented in Figure 1.

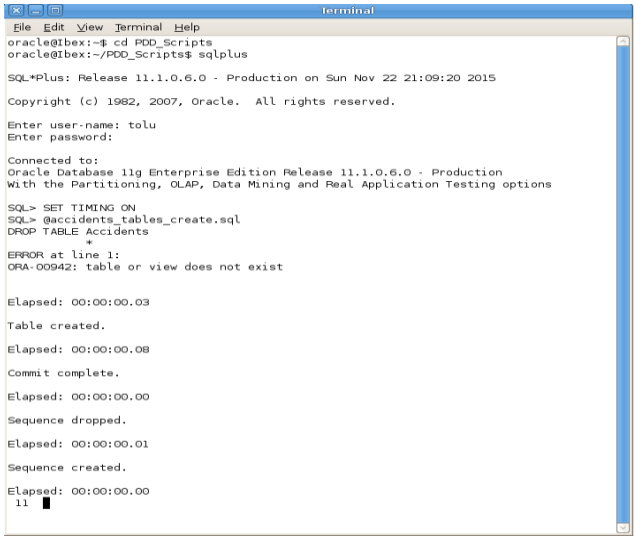

Figure 1 Sample screenshot of create process on Oracle

3.2.2 Accuracy tests (ACT)

Similar procedures were employed for the accuracy tests on both small and large datasets before and after optimisation. The log files created during the load processes were checked to record the total number of correct rows imported when a load query is run. Also, the text files embedded in the timing scripts were checked to record the number of correct rows returned /selected when a read, update, delete and multiple join SQL script was run against the dataset in the database. A sample of the small dataset log file on the Oracle database server is displayed as follows:

Sample of Small Dataset log file on Oracle
*SQL*Loader: Release 11.1.0.6.0 - Production on Tue Nov 24 12:36:09 2015*
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Control File: SMALLDATASET.CON
Data File: SMALLDATASET.csv
Bad File: SMALLDATASET.bad
Discard File: none specified
Table ACCIDENTS:
8075 Rows successfully loaded.
0 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.
Elapsed time was: 00:00:01.98
CPU time was: 00:00:00.03

3.2.3 Throughput tests (THT)

For each of the workload tested above, the timing from the average latency test is divided by the corresponding number of rows obtained from the accuracy tests to calculate the throughput (Number of rows selected/returned per millisecond) of the Oracle database. The results obtained based on the following

formulae were subsequently recorded for the throughput performance tests.

Throughput (ms) = Average Latency (ms)/Accuracy (No of Rows)

3.3 Performance test execution on Cassandra database virtual machine

The CQL queries similar to the SQL queries written for Oracle were run against the small and large datasets in Cassandra and migrated using the ETL processes to test for performance in create and load process, read, update and delete operations after optimisation. In contrast to Oracle, the *cqlsh* utility in Cassandra database was used to create and load data simultaneously. The small and large dataset testing before optimisation was not conducted because Cassandra does not retrieve data unless it is optimised (index key creation). Therefore, only tests for the small and large dataset after optimisation were conducted. The Apache-Cassandra 1.1.6 requires named primary key in the where Clause and it does not support the BETWEEN operator, so few changes were made to the update and delete query. The Apache-Cassandra 1.1.6 does not support joins and views and so few changes were made to the multiple-join statement. Queries were rewritten and saved three times to enable the 3-iterative performance tests. All the queries were run in three iterations and the average value was obtained. The processes involved in the individual performance tests are described as follows.

3.3.1 Average latency tests (ALT)

The processes for the performance tests execution on the small and large datasets are quite similar. For the create and load process, a script was written to create a key-space, use the key-space, create a column family and import the data in .csv format into the Cassandra database using a text-editor and saved in the tests directory. On the server terminal, the script was run and the timing results are displayed on the screen (Figure 2). This process was repeated three times using the same key-space.

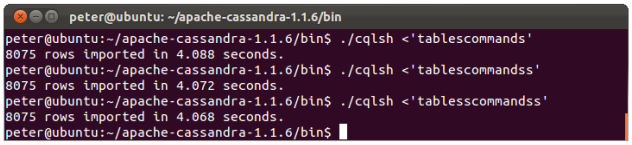

Figure 2 Sample screenshot of create and load process on Cassandra database

For the read, update, delete, and multiple join operations, a CQL script was created to select, update and delete data respectively across the created database and then saved in the bin directory of the database. Thereafter, the scripts were run on the VM terminal using the *cqlsh* utility repeatedly for three times. The obtained timing results are displayed on the screen for recording as shown in Figure 3.

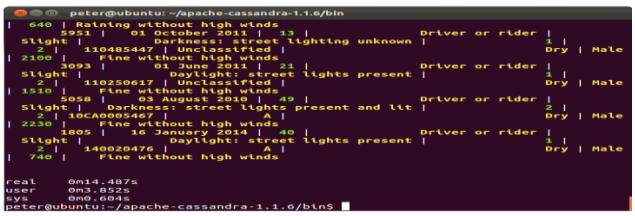

Figure 3 Sample screenshot of read operation on Cassandra database

3.3.2 Accuracy tests (ACT)

Similar procedures were carried out to execute accuracy performance tests on the small and large datasets. For the create and load process, a CQL script saved in the bin directory of Cassandra was written to select accuracy count and re-confirm the results of the timing displayed during the ALT tests above. This script was run against the database and compared with the earlier results. The results were thereafter recorded and was used to determine the accuracy of the database. For read, update, delete, and multiple join operation, a CQL query was written to select accuracy count when any of the query operations is requested. The results which are displayed on the screen were verified and recorded (Figure 4).

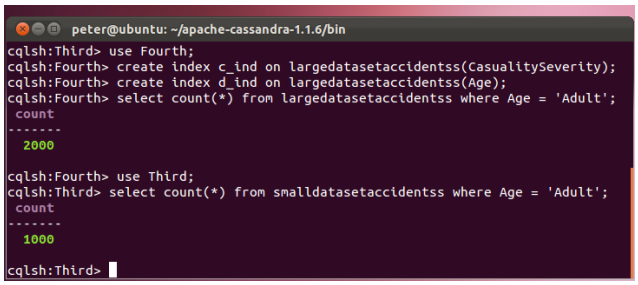

Figure 4 Sample screenshot showing accuracy count of small and large dataset operations on Cassandra

3.3.3 Throughput Tests (THT)

Similarly, for each of the workload tested above, the timing from the average latency test was divided by the corresponding number of rows obtained from the accuracy tests to determine the throughput of the Cassandra database. The results were then recorded for the throughput performance tests based on the following formulae.

Throughput (ms) = Average Latency (ms)/Accuracy (No of Rows)

4. Results and Discussion

An evaluation of the results of the ALT, ACT and THT carried out on the Oracle and Cassandra databases are presented and analysed below.

4.1 Average latency performance test of create and load process on small and large dataset

The ALT values for Oracle were observed to be lower compared with the ALT those of Cassandra, even though Cassandra executed create and load processes together unlike Oracle which executed them separately (Figure 5). Therefore, Oracle was observed to perform better than Cassandra in the creation of tables and loading of large and small structured datasets.

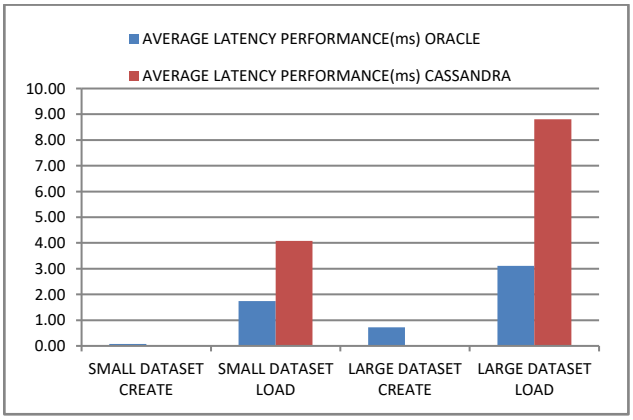

Figure 5 ALT of create and load process on Oracle and Cassandra database

4.2 Average latency performance test of read, update, delete and join operation on large dataset before and after using indexes

The ALT values for Oracle were observed to be lower when compared with those of Cassandra even though Cassandra queries could not be executed without optimisation unlike Oracle which was executed before and after (Figure 6 and 7).

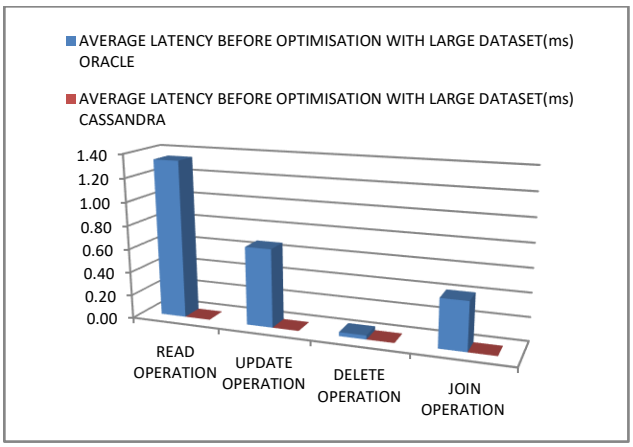

Figure 6 ALT of read, update, delete and multiple join operation on large dataset before using indexes

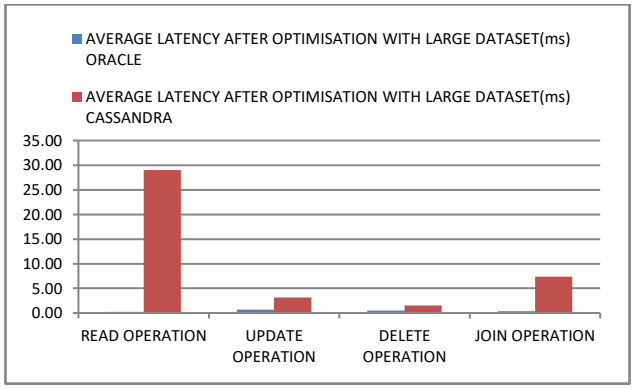

Figure 7 ALT of read, update, delete and multiple join operation on large dataset after using indexes

Therefore, Oracle was observed to perform better than Cassandra in read, update, delete and join operations of queries on the large structured datasets after optimisation

4.3 Accuracy performance tests of create and load process on small and large dataset

The ACT values for Cassandra were examined to be nearly the same when compared with those of Oracle as shown in Table 3. Hence, it could be said that Oracle and Cassandra have approximately equal accuracy performance in the creation and load process of large and small structured datasets.

Table 3 Accuracy Performance Tests for create and load process on small and large datasets before and after optimization

Workload	Oracle	Cassandra
Create tables of small dataset	0	0
Load process of small dataset	8075	8075
Create tables of large dataset	0	0
Load process of large dataset	16147	16148

4.4 Accuracy performance tests of read, update, delete and join operation on large dataset before and after using indexes

The ACT values for Cassandra are observed to be slightly different when compared with the ACT performance values for Oracle in large dataset after optimisation (Table 4). Therefore, Oracle was observed to have an accurate performance in read, delete, update and join of queries on large structured datasets before and after optimisation, while Cassandra was observed to have only an accurate performance in read, update and multiple columns join of queries on large structured datasets after optimisation.

Table 4 Accuracy performance tests of read, update, delete and join operation on large dataset before optimisation

Accuracy before optimisation with large dataset(rows)		
Workload	Oracle	Cassandra
Read operation	14299	0
Update operation	14299	0
Delete operation	14299	0
Multiple table/colomn join operation	16147	0
Accuracy after optimisation with large dataset(rows)		
Workload	Oracle	Cassandra
Read operation	14299	14300
Update operation	14299	2000
Delete operation	14299	16148
Multiple table/colomn join operation	16147	16148

4.5 Throughput performance tests of create and load process on small and large dataset

The THT values for Cassandra are observed to be higher when compared with the THT performance values of Oracle even though Cassandra executed create and load process together unlike Oracle which executed them separately (Figure 8). Hence, Oracle could be said to have better performance than Cassandra in the creation of tables and loading of large and small structured datasets.

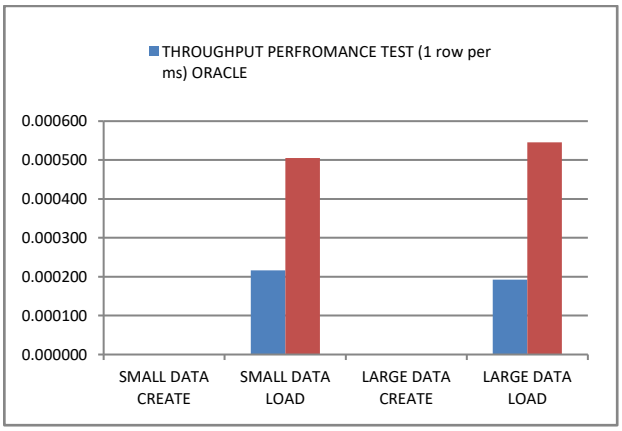

Figure 8 THT performance of create and load process on small and large dataset

4.6 Throughput performance tests of read, update, delete and join operation on large dataset before and after using indexes

The THT values for Cassandra were equally observed to be higher when compared with the THT performance values of Oracle even though Cassandra queries could not be executed without optimisation unlike Oracle which was executed before and after optimization (Figure 9 and 10). This clearly shows that Oracle performed better than Cassandra in read, update, delete and join operations of queries on large structured datasets after optimisation.

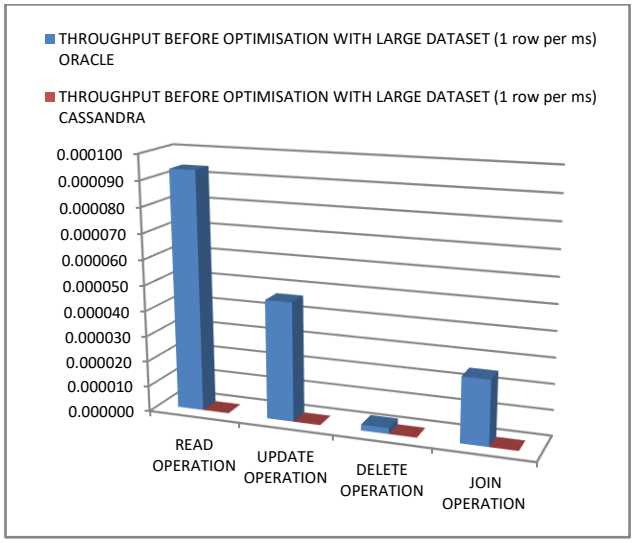

Figure 9 THT performance of read, update, delete and join operation on large dataset before optimisation

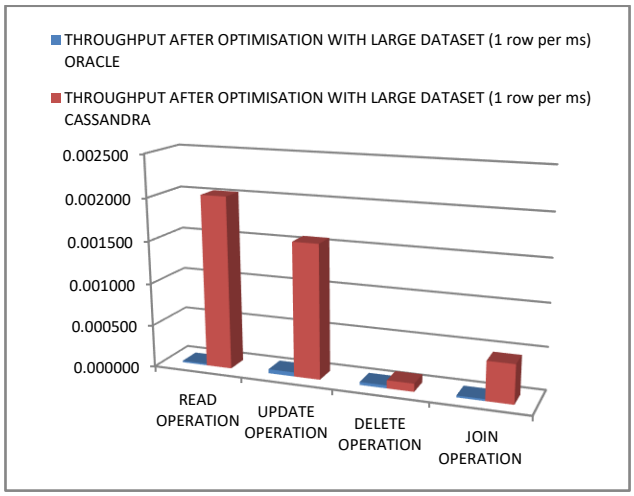

Figure 10 THT performance of read, update, delete and join operation on large dataset after optimisation

Conclusions and Future Works

Based on the results obtained in this study, it could be said that using Oracle as a database for business intelligent systems will provide better performance in terms of latency, throughput, and accuracy when compared with the corresponding performances of Apache Cassandra database. Specifically, Oracle's support for joins and views enables query optimisation for better performance results. Its structured schema enhances accuracy, throughput, and latency during import of data from external sources and CRUD (create, read, update and delete) operations on small and large structured datasets before and after optimisation. These performance outcomes of Oracle may enhance data analytics in the real world of business intelligence. However, Cassandra's support for the creation of column family and imports of data from external sources simultaneously saves time in terms of writing queries; its

processor which supports data indexing before executing queries ensures query optimisation and maintains data quality while its low accuracy in delete operations reduces data reliability and completeness on structured data. These performance outcomes of Cassandra may not provide optimal support for structured data analytics in the real world of business intelligence. Analysing the results presented in Tables 3, 4 and Figures 5, 6, 7, 8, 9,10, it can be observed that Oracle performed better than Cassandra as it exhibited higher throughput and lower latency in the create process, load process, read, update, delete and join operations on small and large structured datasets before and after optimisation. Also, Oracle and Cassandra both have approximately equal accuracy performance as they exhibited almost same results in the create process, load process, read, update and join operations on small and large structured datasets before and after optimisation but a slight difference in delete operations. Therefore, for the development of business intelligent systems in organisations, Oracle database may result in optimal support for small and large structured data analytics than Cassandra database. It is important to note that this study only considered structured datasets in its experiment. As future work, it would be useful to carry out the performance evaluation on unstructured/big data over heterogeneous systems. In addition, it would be relevant to include more metrics such as data consistency, availability, reliability, quality, scalability, and security in the performance tests. Finally, it is important to investigate the cause of the lesser accuracy performance of Cassandra on delete operations.

References

[1] Connolly, Thomas M. and BEGG, Carolyn E. (2015), Database systems: a practical approach to design, implementation, and management, 6th ed., Global ed., *Harlow, Pearson Education.*

[2] gangadharan, G. R. and SWAMI, Sundaravalli N. (2004), Business intelligence systems: design and implementation strategies, In: *Information technology interfaces, 2004, 26th international conference on,* IEEE, 139-144.

[3] Van Der Lans, Rick (2012), *Data Virtualization for business intelligence systems: revolutionizing data integration for data warehouses, Elsevier*.

[4] Olszak, C. M., & Ziemba, E. (2007), Approach to building and implementing business intelligence systems, *Interdisciplinary Journal of Information, Knowledge, and Management*, *2*(1), 135-148.

[5] Chaudhuri, Surajit, Dayal, Umeshwar and Narasayya, Vivek (2011), An overview of business intelligence technology, *Communications of the ACM,* 54 (8), 88-98.

6] ZENG, Li, *et al*. (2006), Techniques, process, and enterprise solutions of business intelligence, In: *Systems, man and cybernetics, 2006, SMC'06. IEEE international conference on,* IEEE, 4722-4726.

[7] Nedelcu, Bogdan (2013), Business Intelligence Systems, *Database Systems Journal,* 4(1), 12-20

[8] Elbashir, Mohamed Z., Collier, Philip A. and Davern, Michael J. (2008), Measuring the effects of business intelligence systems: The relationship between business process and organizational performance, *International journal of accounting information systems,*9 (3), 135-153.

[9] Moniruzzaman, ABM and Hossain, Syed Akhter (2013), NoSQL database: New era of databases for big data analytics-classification, characteristics and comparison, *arXiv preprint arXiv:1307.0191.*

[10] Van der Veen, Jan Sipke, Van Der Waaij, Bram and MEIJER, Robert J. (2012), Sensor data storage performance: SQL or NoSQL, physical or virtual, In: *Cloud computing (CLOUD), 2012 IEEE 5th international conference on* IEEE, 431-438.

[11] Grolinger, Katarina, *et al*. (2013), Data management in cloud environments: NoSQL and NewSQL data stores, *Journal of cloud computing: Advances, systems and applications,* 2 (1), 1.

[12] Oracle (2015), *Managing Oracle Enterprise Manager Database Control,* [Online] California, USA, Oracle Corporation, Last accessed on 01 Dec. 2015 at: https://docs.oracle.com/cd/E11882_01/server.112/e25494/dbcontrol.htm#ADMIN13401

[13] DATASAX (2015). *About Apache Cassandra.* [Online] California, USA, Datasax Inc, Last accessed on 01 Dec. 2015 at: http://docs. datastax.com/en/ cassandra/2.1/cassandra/gettingStartedCassandraIntro.html

[14] LEE, Ken Ka-Yin, TANG, Wai-Choi and CHOI, Kup-Sze (2013), Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage, *Computer methods and programs in biomedicine,* 110 (1), 99-109.

[15] HECHT, Robin and Jablonski, Stefan (2011), Nosql evaluation, In: *International conference on cloud and service computing,* IEEE, 336-341.

[16] RABL, Tilmann, *et al*. (2012), Solving big data challenges for enterprise application performance management, *Proceedings of the VLDB endowment,* 5 (12), 1724-1735.

[17] Tudorica, Bogdan G. and Bucur, Cristian (2011), A comparison between several NoSQL databases with comments and notes, In: *2011 RoEduNet international conference 10th edition: Networking in education and research,* IEEE, 1-5.

[18] LI, Yishan and Manoharan, Sathiamoorthy (2013), A performance comparison of SQL and NoSQL databases, In: *Communications, computers and signal processing (PACRIM), 2013 IEEE pacific rim conference on,* IEEE, 15-19.