SQL Database with physical database tuning technique and NoSQL graph database comparisons

Wisal Khan
School of Computer and Technology
Anhui University
Hefei 230039
Peoples Republic of China
Email: wisalkhan52@gmail.com

Waqas Ahmad
School of Computer and Technology
Anhui University
Hefei 230039
Peoples Republic of China
Email: waqasasad007@yahoo.com

Bin Luo
School of Computer and Technology
Anhui University
Hefei 230039
Peoples Republic of China
Email: luobin@ahu.edu.cn

Ejaz Ahmed
National University of Computer and Emerging Sciences
Islamabad
Pakistan
Email: ejaz.ahmed@nu.edu.pk

Abstract—**Relational databases are used in many organizations of various natures from last three decades such as Education, health, businesses and in many other applications. SQL databases are designed to manage structured data and show tremendous performance. Atomicity, Consistency Isolation, Durability (ACID) property of Relational databases is used to manage data integrity and consistency. Physical database techniques are used to increase the performance of relational databases. Tablespaces also called subfolder is one of the physical database technique used by Oracle SQL database. Tablespaces are used to store the data logically in separate data files. Now-a-days huge amount and varied nature (unstructured and semi structured) of data is generated by the various organizations i.e. videos, images, blogs etc. This large amount of data is not handled by the SQL databases efficiently. NoSQL databases are used to process and analyze the large amount of data efficiently. Four different types of NoSQL databases are used in the industry according to the organization requirement. In this article, first, we do the physical database tuning of the Oracle Relational database and then compared with NoSQL Graph database. Relational database performance is increased up to 50% due to physical database tuning technique (Tablespaces). Besides, physical database tuning approach of relational database NoSQL graph database performed better in all our proposed scenarios.**

Keywords: Relational databases, Physical database tuning, tablespaces, NoSQL Graph database

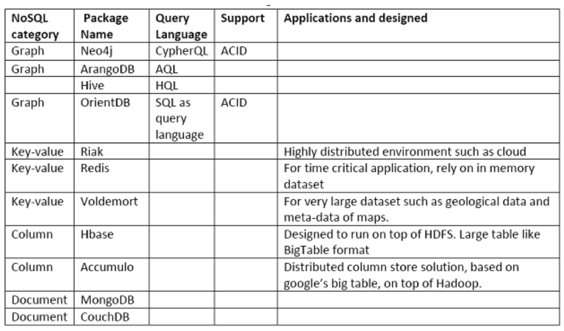
I. Introduction

In Industry, various kind of data is generated such as semi structured and unstructured data and expended rapidly. The storage of such a large amount of data is not the only issue but also how to access this amount of data quickly and accurately. From last three decades, such large amount of data is being handled and managed by the traditional databases. The ACID property of SQL database is used to keep the data consistent and efficient to extract meaningful knowledge. Structured data is manipulated through SQL declarative language of relational databases. The data consistency is the main theme of the SQL databases and can process the data at certain limit [1]. Organizations are required to increase their system capacity such as RAM, Disk; optimized method of accessing data etc. manage large datasets through relational databases. The systems have also limited capacity.

In Relational databases, tables are used to store the data. This way of storage to read and understand the data is not the suitable method [7]. Data inconsistency cannot be eliminated to index cross reference data. Hence, tables are used to store data in the relational databases. Theses tables are easy to understand and readable. Whenever, to eliminate data redundancy and inconsistencies from these tables through normalization techniques (1NF, 2NF, and 3NF), start referencing and generating referential integrity. The use of without complicated joins and queries it is difficult to understand and maintain the data due to these referential integrity constraints. SQL databases have many techniques to access and search data quickly and efficiently like indexing, sub databases, partitioning and query optimization. Query time index lookup is performed for each and every join in the bunch of joins scenario and work well. The mentioned method is become more expensive when we have 20 or more joins in a query. It becomes much more expensive as the dataset is being increases with the passage of time.

SQL databases are vertical scalable databases and can process certain amount of data. While many organizations, are creating large amount of varied and connected nature of data. New technology, approaches and storage techniques are introduced to manage spare, large and connected datasets. Therefore, NoSQL databases are used to process such huge amount data efficiently. NoSQL databases are horizontal scalable databases. BASE (Basically Available State and Eventually Consistent) is the main property of NoSQL databases [1]. Graph database is one of the NoSQL databases type. It is used to process large amount of connected datasets efficiently.

Specifically, Graph databases are created and designed for connected nature of data. They are used in many applications LinkedIn, Facebook, Amazon and many more. The literature describes that NoSQL graph performed better than relational databases [8, 9]. Our previous paper [8] also proved that NoSQL graph database (Neo4j) performed well than SQL database (Oracle 11g).



NoSQL category	Package Name	Query Language	Support	Applications and designed
Graph	Neo4j	CypherQL	ACID	
Graph	ArangoDB	AQL		
	Hive	HQL		
Graph	OrientDB	SQL as query language	ACID	
Key-value	Riak			Highly distributed environment such as cloud
Key-value	Redis			For time critical application, rely on in memory dataset
Key-value	Voldemort			For very large dataset such as geological data and meta-data of maps.
Column	Hbase			Designed to run on top of HDFS. Large table like BigTable format
Column	Accumulo			Distributed column store solution, based on google's big table, on top of Hadoop.
Document	MongoDB			
Document	CouchDB			

Fig. 1. demonstrates various NoSQL databases packages and their characteristics.

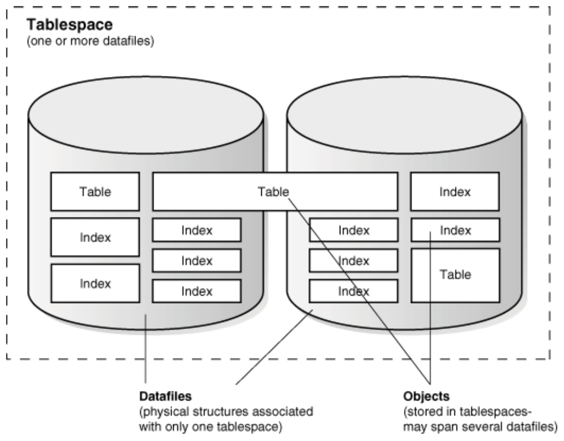


Fig. 2. Data files and Tablespaces.

NoSQL databases have four different types and each one of them is used according to the organization needs and requirements [10]. The names are: Key Value, Document, Column and Graph databases. 1) Key-Value: hash table is the core technique used by the key value database. Hash table uses unique key and pointer to find the particular item. The large number of records is efficiently processed by Hash table. 2) Document databases: store data as key/value. They are different from Key-value databases. Both the key and document contents are used to search a particular document. XML, JSON (Java Script Object Notation) and BSON (Binary JSON) forms are used to store a document. Popular examples of document databases are MongoDB and CouchDB. 3) Wide-Column databases: is based on hybrid architecture. It means these databases used the techniques and approaches of relational databases and Key-value databases are used to store schema. They are mainly used in cluster environment for distributed data. Hbase, Cassandra and Accumulo are the mentioned examples. 4) The main theme of this article is the Graph database. Graph databases are mainly used for connected nature of data. They are used to process and analyze connected data efficiently. The main strength of graph databases is not only stored the objects but also stored the relationship among the objects. The examples of graph databases are Neo4j, Pregel, ArrangoDB and OrientDB. The various packages of NoSQL databases and their characteristics are described by figure 1.

A. Physical database design

Physical design or schema tuning of any database management (DBMS) system consists of the process in which logical data has transformed into the physical structure [2]. Frontier design is a method that presents benets of virtualization in database environment. Authors states the different problems of tuning related to the virtualization. They proposed a cost model in this respect to solve tuning problems [6]. Some tools have automatic support for tuning called self-tuning tools. Automated physical design is an approach to enhance the performance of physical database design. Performance greatly depends upon the workload [5].

1) Oracle 11g Tablespaces: Oracle 11g, Relational database management system (RDBMS) stores the data logically in tablespaces while data files associated with the corresponding tablespace are used to store the data physically [3,4]. Oracle 11g stores all the user s table data in a single tablespace in default conguration. To improve the performance of Oracle 11g database, we can create separate tablespaces for each individual schema and tables. Figure 2 represents the Tablespaces and Data files.

Data files are distributed into small chunks. These small chunks are known as extents and default size of a single extent is 64K. Further then extents are divided into small blocks and are called data blocks or oracle blocks. The block size can be 2K, 4K, 8K, 16K and 32K. Default block size is 8K. The block size can affect the database performance because in the memory database buffer cache size is same as database block size. According to the database type we can choose the appropriate block size. For Data warehouse type of database, we normally keep block size large while for OLTP type of database the block size is always small. Extents are used for allocation and for deallocation and can fix its size at the time of table creation or tablespace creation. Extents are the minimum unit of space allocation while blocks are the minimum unit of Input/Output. One extent can be assigned to one object. Whenever, all the blocks of an extent filled of a particular object then next available extent is assigned to that particular object and so on. In our study we choose the extent size 1024K and the block size 8K. Figure 3 describes the data files, extents and data blocks.

The rest of the paper is organized as follow. Section II presents related work. The proposed method and experiments is described by section III and section IV respectively. Section V describes result and evaluation while conclusion is given in section VI.

II. Related Word

A. Comparisons of SQL Databases and NoSQL Databases

Users of SQL databases [1] are required to handle large amount of varied nature of data by increasing the system

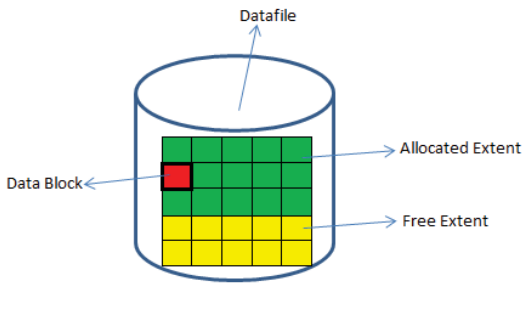


Fig. 3. Data file, Extents and Data Blocks.

capacity like CPU, RAM etc. Relational databases are vertical scalable databases and can process the data at certain limit. NoSQL databases are horizontal scalable databases and can process huge amount of data efficiently. Trustful transactions, better data integrity and security are the main focus and strength of the relational databases. BASE approach is used by the NoSQL databases and can easily manage various forms of massive data. Handling big data through NoSQL databases required lower cost and minimum overhead. The new server can be added easily in the cluster environment due to horizontal scalable characteristics of NoSQL databases. In the cluster commodity hardware are used to store big data.

Eric Brewer [11] presented the CAP theorem and stand for Consistency, Availability and Partition Tolerance. Amazon like large companies implemented and adopted the CAP theorem. In CAP, consistency describes how to bring the system in a consistent state after performing some writes operations by the system. Availability describes, after performing the writes operations, the updated data must always be highly available to the users. Partition tolerance describes if the data is spread and distributed over the various nodes in a network the system must responded and be able to continue its operations. The C (Consistency) and P (Partition Tolerance) of CAP theorem is used by the SQL databases. NoSQL databases uses the A (Availability) and P (Partition Tolerance) of the CAP theorem. A and P of the CAP theorem is used by the Amazon Dynamo.

In Relational databases [7], tables are used to store the data. The primary key concept is used to prevent repetitions. Hence, data is always in a consistent state and reliable. The approach and technique of NoSQL databases is different. The data is distributed on different nodes to process, analyze and evaluate the data and performed the task fast and efficiently. The rigid schema structure of RDBMS with the passage of time becomes more mature. In the mature rigid schema, to bring any changes if necessary is very hard but possible. NoSQL databases do not follow the rigid schema structure. The schema of NoSQL databases is flexible and developed gradually. NoSQL databases can also handle the NULL values problems of stored data in RDBMS.

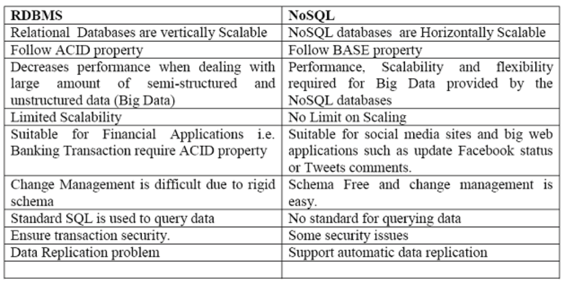
Directed acyclic graph [9], is the recommended storage structure for data provenance information. Whether or not, the MySQL relational database and Neo4j NoSQL graph database would be the reliable and the appropriate choice, for the developing of data provenance system. Graph is the fundamental data structure in computer science and in software engineering. There are many kinds of graph software which stores and query the graph. The counter examples of graph databases are LinkedIn, Facebook and Google. These social applications used graph databases to store their connected and large amount of data. Modeling object interactions can be easily managed through graph databases.

We Compared [8], the Oracle relational database and NoSQL graph database using optimized queries. Our experiment has shown whenever data becomes more and more connected (large number of joins) and large in size, relational databases shown worse performance than NoSQL graph database. Relational database (Oracle 11g Enterprise Edition) uses constraints, indexes and does not store any relationship information. While NoSQL graph database stores relationship information among various nodes. Graph database (Neo4j 3.0.3) uses native graph storage. Native graph is optimized and designed for storing and managing graph. NoSQL graph database uses index-free adjacency. The connected nodes physically points to each other in the graph database due to index-free adjacency characteristics.

They did comparison [12] of relational and NoSQL databases. They evaluate and analyze both nature of databases by using various characteristics, approaches, dimensions and techniques. They conclude that not all NoSQL databases performed better than SQL databases. Both kinds of databases have their own advantages and disadvantages. NoSQL databases have the lack of standard query language and do not have many users. Relational databases have standard query language (SQL). NoSQL databases can process huge amount and varied nature of data easily. The same is not true for relational databases. Relational databases have better data consistency than NoSQL databases. NoSQL databases have better scalability than SQL databases. They conclude that according to the organizations need and requirement the appropriate SQL or NoSQL databases can be selected.

In [13], they compared the performance of Microsoft SQL server relational database and MongoDB NoSQL database. MongoDB performed well than SQL server in simple query, insert and update. SQL server performed better for aggregate queries and queries with secondary attributes.

MySQL relational database and MongoDB were compared [14]. Their experiments have shown that MongoDB performance is very well than MySQL database. MongoDB performed better in the insertion scenario when large number of records is inserted in both databases. NoSQL databases are the appropriate choice for developing big data applications due to its dynamic schema characteristics.



RDBMS	NoSQL
Relational Databases are vertically Scalable	NoSQL databases are Horizontally Scalable
Follow ACID property	Follow BASE property
Decreases performance when dealing with large amount of semi-structured and unstructured data (Big Data)	Performance, Scalability and flexibility required for Big Data provided by the NoSQL databases
Limited Scalability	No Limit on Scaling
Suitable for Financial Applications i.e. Banking Transaction require ACID property	Suitable for social media sites and big web applications such as update Facebook status or Tweets comments.
Change Management is difficult due to rigid schema	Schema Free and change management is easy.
Standard SQL is used to query data	No standard for querying data
Ensure transaction security.	Some security issues
Data Replication problem	Support automatic data replication

Fig. 4. Comparisons of relational and NoSQL databases.

The author did comparative study of SQL and NoSQL databases [15]. SQL Server and RavenDB performance were noticed and analyzed. RavenDB performance is 50% better than SQL Server database. The main features of both database is given in the Figure 4.

B. Physical Database Design and Tuning Approaches

Physical design or schema tuning [2] of any database management (DBMS) system consists of the process in which logical data has transformed into the physical structure. For example, schema design process starts from the entity relationship diagram (ERD) and it includes the quality selection of suitable data storage approach and index usage. When dealing with large set of data, design of database is very important. For example, well defined schemas into tables, table spaces, partitions, projections will definitely results the query in less time while the other case where schema design is not appropriate according to the workload this will give result in more time as well as data loading issues may concerned. To manage large sets of data, effective database tuning is required. Database tuning is certainly a difficult task especially when DBA deals with large number of data. There are many approaches and tools which perform database tuning. For example, database tuning advisor (DTA) is the most famous tools.

Author presents [5] a method to design physical database configuration on the basis of or using principle of B+ tree. Appropriate storage structure will reduce cost of storage and increase performance. B trees have great importance in searching of database. These are implemented in different applications for increasing performance. Authors proposed a method on the basis of B+ tree principle instead of other data structure. Schema tuning or physical database design tuning is important as the workload increases performance will degraded if appropriate settings will not be implemented.

Frontier design is a method [6] that presents benefits of virtualization in database environment. Authors states the different problems of tuning related to the virtualization. They proposed a cost model in this respect to solve tuning problems.

The Authors proposed a technique named tuning ball [16] that consist of physical design, query optimization and columnar store database. They conducted an experiment to check the effectiveness of tuning ball. In their experiment they first execute the dataset on default settings of Oracle 11g to measure the performance. Then in second fold they enhanced the physical design by adding advance tuning parameters and compare it with the performance of oracle 11g results shows that their proposed technique tuning ball is more effective for row store databases. As their technique contains advance tuning parameters it is proved that for large number of records performance can be enhanced by using tuning ball.

The [17], describes a systematic research in the domain of database architectures. Authors research the most recent type of database architecture commonly called column oriented or column stores. Basically this is a survey paper targeting the research from 1970s to till. Initially authors briey explain what does mean by horizontal organization and vertical organization of data. For example design of row stores is data is stored row-wise, one row follows its precursor. Whole table is stored in disk page or file and file contain multiple rows. While vertical organization of data stores is column wise.

Oracle stores information sensibly in tablespaces [3] and physically in data files connected with the relating tablespace. [18], An Oracle database comprises of at least one legitimate stockpiling units called tablespaces, which all things considered store the greater part of the databases information. Each tablespace in an Oracle database comprises of at least one document called data files, which are physical structures that fit in with the working framework in which Oracle is running. A databases information is by and large put away in the data files that constitute each tablespace of the database. For instance, the most straightforward Oracle database would have one tablespace and one data file. Another database can have three tablespaces, each comprising of two data files (for an aggregate of six data files).

III. Proposed Method

The proposed methodology consists of SQL database (ORACLE) with physical database tuning technique called tablespaces and NoSQL graph database. The Oracle relational database performance can be increased with the physical database tuning. In [8], we compared the same databases without physical database tuning technique of relational database. The NoSQL graph database performed better than Oracle SQL database with its default configuration. By using physical database tuning technique, we can increase the performance of Oracle relational database. Therefore, we will use the same MedCare dataset of our previous paper and will notice the performance of both databases for a standalone system. Our desired research method is described by figure 5.

IV. Experiments

We used the medical dataset a Medcare case study for our desired experiments to evaluate and analyze the proposed research process flow. We used Neo4j (3.03) community edition and Oracle 11g enterprise edition for our desired experiments.

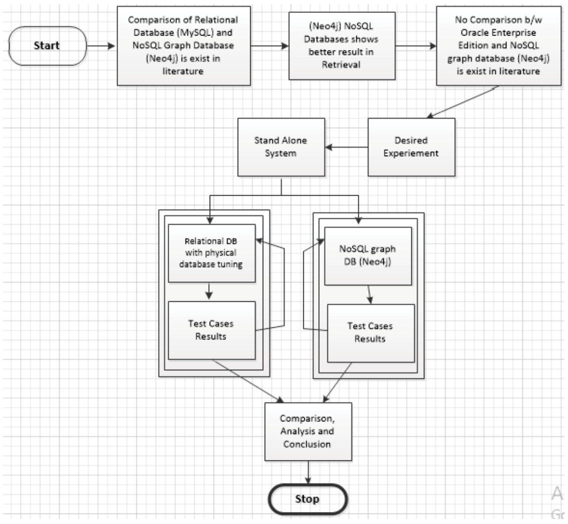

Start
Comparison of Relational Database (MySQL) and NoSQL Graph Database (Neo4j) is exist in literature
(Neo4j) NoSQL Databases shows better result in Retrieval
No Comparison b/w Oracle Enterprise Edition and NoSQL graph database (Neo4j) is exist in literature
Stand Alone System
Desired Experiement
Relational DB with physical database tuning
Test Cases Results
NoSQL graph DB (Neo4j)
Test Cases Results
Comparison, Analysis and Conclusion
Stop

Fig. 5. Proposed Method.

S#	Object Name	Number of Records	File Size in MB
1	Patient	27952	04 MB
2	Dependent	19036	2 MB
3	Patient_Visit	625721	320 MB
4	Patient_IssueMed	869666	80 MB

Fig. 6. Object Name, size and Number of records [8].

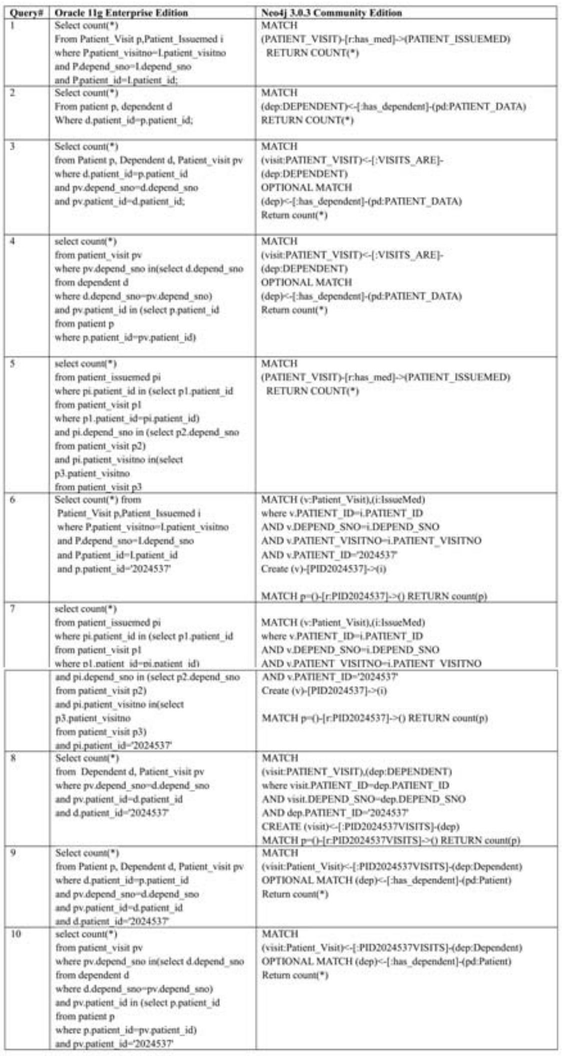
The effectiveness of both databases is noticed for our desired queries.

The healthcare case study Medcare schema consists of the following primary objects: Patient Table, Dependent Table, Patient Visit Table, Patient Issued Medicine table, Patient History table, Medical Staff table and Patient Appointment table. The data in the mentioned tables and the Medcare dataset have been updated yearly. The total number of records in each schema table is given in the figure 6.

We use the same objective benchmark of both databases as we used in [8]. The following properties are used by the objective benchmark:

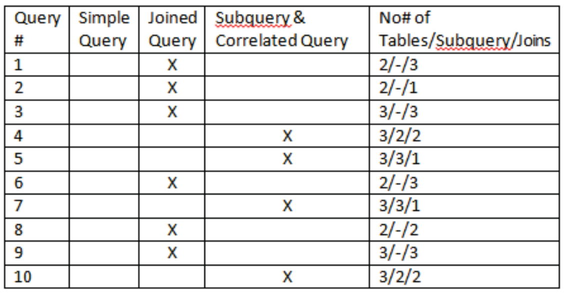
- Characteristics of Scalability
- Requirements of Disk Space
- Predefined Queries Set

The detail analysis of both databases is based on the given ten queries. These ten queries are performed and executed on the mentioned database objects in the desired environment. From query 1 to query 5 are already used in [8]. The desired ten queries are given in Figure 7. Figure 8, represent each query characteristics and complexity.



Query#	Oracle 11g Enterprise Edition	Neo4j 3.0.3 Community Edition
1	Select count(*) From Patient_Visit p,Patient_Issuemed i where P.patient_visitno=I.patient_visitno and P.depend_sno=I.depend_sno and P.patient_id=I.patient_id;	MATCH (PATIENT_VISIT)-[r:has_med]->(PATIENT_ISSUEMED) RETURN COUNT(*)
2	Select count(*) From patient p, dependent d Where d.patient_id=p.patient_id;	MATCH (dep:DEPENDENT)<-[:has_dependent]-(pd:PATIENT_DATA) RETURN COUNT(*)
3	Select count(*) from Patient p, Dependent d, Patient_visit pv where d.patient_id=p.patient_id and pv.depend_sno=d.depend_sno and pv.patient_id=d.patient_id;	MATCH (visit:PATIENT_VISIT)<-[:VISITS_ARE]- (dep:DEPENDENT) OPTIONAL MATCH (dep)<-[:has_dependent]-(pd:PATIENT_DATA) Return count(*)
4	select count(*) from patient_visit pv where pv.depend_sno in(select d.depend_sno from dependent d where d.depend_sno=pv.depend_sno) and pv.patient_id in (select p.patient_id from patient p where p.patient_id=pv.patient_id)	MATCH (visit:PATIENT_VISIT)<-[:VISITS_ARE]- (dep:DEPENDENT) OPTIONAL MATCH (dep)<-[:has_dependent]-(pd:PATIENT_DATA) Return count(*)
5	select count(*) from patient_issuemed pi where pi.patient_id in (select p1.patient_id from patient_visit p1 where p1.patient_id=pi.patient_id) and pi.depend_sno in (select p2.depend_sno from patient_visit p2) and pi.patient_visitno in(select p3.patient_visitno from patient_visit p3	MATCH (PATIENT_VISIT)-[r:has_med]->(PATIENT_ISSUEMED) RETURN COUNT(*)
6	Select count(*) from Patient_Visit p,Patient_Issuemed i where P.patient_visitno=I.patient_visitno and P.depend_sno=I.depend_sno and P.patient_id=I.patient_id and p.patient_id='2024537'	MATCH (v:Patient_Visit),(i:IssueMed) where v.PATIENT_ID=i.PATIENT_ID AND v.DEPEND_SNO=i.DEPEND_SNO AND v.PATIENT_VISITNO=i.PATIENT_VISITNO AND v.PATIENT_ID='2024537' Create (v)-[PID2024537]->(i) MATCH p=()-[r:PID2024537]->() RETURN count(p)
7	select count(*) from patient_issuemed pi where pi.patient_id in (select p1.patient_id from patient_visit p1 where p1.patient_id=pi.patient_id) and pi.depend_sno in (select p2.depend_sno from patient_visit p2) and pi.patient_visitno in(select p3.patient_visitno from patient_visit p3) and pi.patient_id='2024537'	MATCH (v:Patient_Visit),(i:IssueMed) where v.PATIENT_ID=i.PATIENT_ID AND v.DEPEND_SNO=i.DEPEND_SNO AND v.PATIENT_VISITNO=i.PATIENT_VISITNO AND v.PATIENT_ID='2024537' Create (v)-[PID2024537]->(i) MATCH p=()-[r:PID2024537]->() RETURN count(p)
8	Select count(*) from Dependent d, Patient_visit pv where pv.depend_sno=d.depend_sno and pv.patient_id=d.patient_id and d.patient_id='2024537'	MATCH (visit:PATIENT_VISIT),(dep:DEPENDENT) where visit.PATIENT_ID=dep.PATIENT_ID AND visit.DEPEND_SNO=dep.DEPEND_SNO AND dep.PATIENT_ID='2024537' CREATE (visit)<-[:PID2024537VISITS]-(dep) MATCH p=()-[r:PID2024537VISITS]->() RETURN count(p)
9	Select count(*) from Patient p, Dependent d, Patient_visit pv where d.patient_id=p.patient_id and pv.depend_sno=d.depend_sno and pv.patient_id=d.patient_id and d.patient_id='2024537'	MATCH (visit:Patient_Visit)<-[:PID2024537VISITS]-(dep:Dependent) OPTIONAL MATCH (dep)<-[:has_dependent]-(pd:Patient) Return count(*)
10	select count(*) from patient_visit pv where pv.depend_sno in(select d.depend_sno from dependent d where d.depend_sno=pv.depend_sno) and pv.patient_id in (select p.patient_id from patient p where p.patient_id=pv.patient_id) and pv.patient_id='2024537'	MATCH (visit:Patient_Visit)<-[:PID2024537VISITS]-(dep:Dependent) OPTIONAL MATCH (dep)<-[:has_dependent]-(pd:Patient) Return count(*)

Fig. 7. Executed Queries.



Query #	Simple Query	Joined Query	Subquery & Correlated Query	No# of Tables/Subquery/Joins
1		X		2/-/3
2		X		2/-/1
3		X		3/-/3
4			X	3/2/2
5			X	3/3/1
6		X		2/-/3
7			X	3/3/1
8		X		2/-/2
9		X		3/-/3
10			X	3/2/2

Fig. 8. Queries Characteristics and Complexity level.

V. Results and Evaluation

A. Configuration of Oracle 11g with physical database tuning technique and Neo4j on Local System

In [8] Neo4j graph database performed better than Oracle relational database for the first 5 queries. Table 1; represent the results of ten queries of both databases and SQL database is without physical database tuning technique. First five queries are already used in [8] of table 1.

Query#	Oracle 11g	Neo4j3.0.3
1	4.515 Sec	0.346 Sec
2	0.172 Sec	0.0216 Sec
3	3.531 Sec	0.0452 Sec
4	3.469 Sec	0.0452 Sec
5	10.391 Sec	0.346 Sec
6	3.797 Sec	2.088 Sec
7	6.704 Sec	2.088 Sec
8	4.094 Sec	1.482 Sec
9	3.438 Sec	0.124 Sec
10	3.313 Sec	0.124 Sec

TABLE I
QUERIES RESULTS OF NEO4J AND ORACLE 11G WITHOUT PHYSICAL DATABASE TUNING.

Query#	Oracle 11g	Neo4j3.0.3
1	0.953 Sec	0.346 Sec
2	0.091 Sec	0.0216 Sec
3	0.922 Sec	0.0452 Sec
4	0.769 Sec	0.0452 Sec
5	9.359 Sec	0.346 Sec
6	2.823 Sec	2.088 Sec
7	5.219 Sec	2.088 Sec
8	3.189 Sec	1.482 Sec
9	2.057 Sec	0.124 Sec
10	2.465 Sec	0.124 Sec

TABLE II
QUERIES RESULTS OF NEO4J AND ORACLE 11G WITH PHYSICAL DATABASE TUNING.

The table I, describes that Neo4j performed well in all our desired scenarios. By using, physical database tuning technique we can improve the time of each query of figure 7. Table II, demonstrates the result of the same ten queries of both databases. This time physical database tuning technique (Tablespaces) is used for Oracle relational database.

Table II, describes that the queries execution time of Oracle 11g has been improved from the queries execution time of Oracle 11g of Table 1 due to Oracle 11g tablespaces. Tablespaces is one of the physical database tuning techniques. Besides the physical database tuning technique, the Neo4j NoSQL graph database performed much better than Oracle 11g relational database. The figure 8 and figure 9 are the graphical representation of tables I and II respectively. X-axis represents the executed queries while Y-axis represents time in seconds of the executed queries.

The average times taken by the executed queries in our three proposed experiments are given in table III.

Table III, describes the average of a relational databases has been decreased from 4.34 to 2.78 due to physical database tuning technique (Oracle 11g tablespaces). Almost, tablespaces increased approximately 50% performance of a relational database. Besides, the tablespaces the Neo4j performed better in all our proposed scenarios the Oracle 11g relational database.

Desired Environment	Average (Arithmetic Mean)
SQL Database without physical database tuning	4.34
SQL Database with physical database tuning	2.78
Neo4j NoSQL graph database	0.67

TABLE III
AVERAGE OF THE PROPOSED EXPERIMENTS.

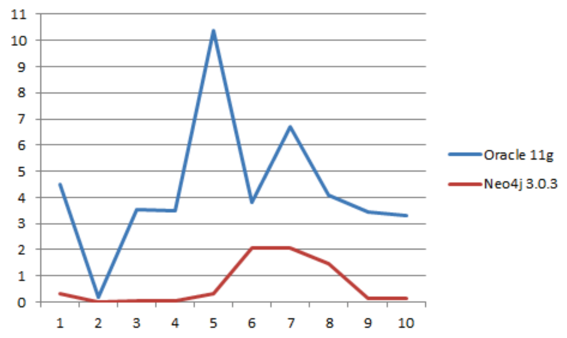


Fig. 9. SQL database without Physical database tuning technique and Neo4j. X-axis represents executed queries and Y-axis represents time in seconds taken by each executed query

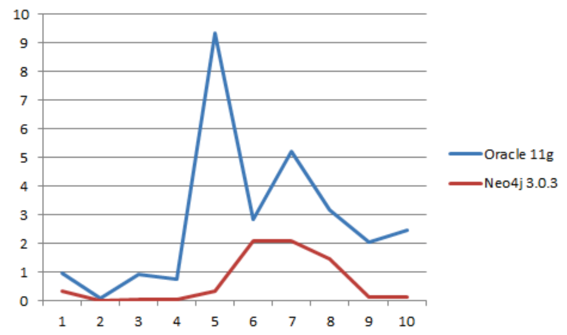


Fig. 10. SQL database with Physical database tuning technique and Neo4j. X-axis represents executed queries and Y-axis represents time in seconds taken by each executed query

VI. CONCLUSION

The graph database Neo4j performed well for large and connected nature of data. The performance of Neo4j graph database becomes much better as the data size and number of joins increases. The physical database tuning technique tablespaces increases the execution time of each executed query of Oracle 11g SQL database. The strength of the Neo4j graph database is just not stored the database objects but also stored their relationship among them. While Oracle 11g relational database computes the number of joins at query execution time. More joins leads to more complexity and more complexity leads to more query execution time in Oracle 11g relational database. Approximately 50% performance increases has been shown in relational databases by physical database tuning technique (Tablespaces) than without physical database tuning technique. Therefore, our designed experiment has shown that besides the physical database tuning technique tablespace of Oracle 11g, Neo4j performance is better in all scenarios.

This work can be considered for future work and can enhance the comparative study of both databases. There are various physical database tuning techniques which can improve the performance of SQL databases like partitioning and query optimization etc.

REFERENCES

[1] Oussous, A., F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih (2015). Comparison and classication of nosql databases for big data. In Proceedings of International Conference on Big Data, Cloud and Applications.

[2] Raizman, A., A. Marathe, D. Milton, D. Sonkin, L. Kollar, M. Sarnowicz, M. Syamala, R. Duddupudi, S. Agrawal, S. Chaudhuri, et al. (2004). Database tuning advisor.

[3] Li, Q. and H. Xu (2009). Research on the backup mechanism of oracle database. In Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on, Volume 2, pp. 423426. IEEE.

[4] https://docs.oracle.com/cd/B28359 01/server.111/b28318/physical.htm#CNCPT401

[5] Malik, T., X. Wang, R. Burns, D. Dash, and A. Ailamaki (2008, April). Automated physical design in database caches.

[6] Soror, A. A., A. Aboulnaga, and K. Salem (2007, April). Database virtualization: A new frontier for database tuning and physical design.

[7] Zafar, R., M. F. Zuhairi, E. Ya, and H. Dao. Big data: The nosql and rdbms review.

[8] Wisal Khan, Ejaz Ahmed, Waseem Shahzad,(July 2017). Predictive Performance Comparison Analysis of Relational & NoSQL Graph Database, International Journal of Advanced Computer Science and Application (IJACSA), ISI Indexed, ISSN: 2156-5570.

[9] Vicknair, C., M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins(2010). A comparison of a graph database and a relational database: a data provenance perspective. In Proceedings of the 48th annual Southeast regional conference, pp. 42. ACM.

[10] Moniruzzaman, A. and S. A. Hossain (2013). Nosql database: New era of databases for big data analytics-classication, characteristics and comparison. arXiv preprint arXiv:1307.0191.

[11] Strauch, C., U.-L. S. Sites, and W. Kriha (2011). Nosql databases. Lecture Notes, Stuttgart Media University.

[12] Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B., & Ismaili, F (2018). Comparison between relational and NOSQL databases.

[13] Z. Parker, S. Poe, S. V. Vrbsky,(April 4-6, 2013). Comparing NoSQL MongoDB to an SQL DB, In Proceedings of the 51st ACM Southeast Conference (ACMSE '13). ACM, New York, NY, USA, Article 5, 6 pages.

[14] C. Gyrdi, R. Gyrdi, G. Pecherle, A. Olah,(11-12 June 2015). A Comparative Study: MongoDB vs. MySQL, Conference: The 13th International Conference on Engineering of Modern Electric Systems, pp. 1-6. Oradea.

[15] Fraczek K., Plechawska-Wojcik M.,(May 30 - June 2, 2017). Comparative Analysis of Relational and Non-relational Databases in the Context of Performance in Web Applications, Proceedings, 13th International Conference, Beyond Databases, Architectures and Structures (BDAS 2017), pp. 153-164, Ustro, Poland.

[16] Saba Luqman and Ejaz Ahmed(2016). Systematic Mapping: Database Tuning Progress in a Decade, Journal of Basic and Applied Scientic Research (JBASR), ISSN: 2090-24x, Indexed in Copernicus, Vol. 6(11), pp. 15-25.

[17] Svensson, P., P. Boncz, M. Ivanova, M. Kersten, N. Nes, D. Rotem, and A. Shoshani (2010). Emerging database systems in support of scientic data. Scientic Data Management: Challenges Technology and Deployment.

[18] Raizman, A., A. Marathe, D. Milton, D. Sonkin, L. Kollar, M. Sarnowicz, M. Syamala, R. Duddupudi, S. Agrawal, S. Chaudhuri, et al. (2004). Database tuning advisor.