IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

Graph Database: An Alternative for Relational Database

H.R. Vyawahare[1*], P.P. Karde[2], V.M. Thakare[3]

[1,3]Post Graduate Department of Computer Science, SGBAU, Amravati, India
[2]Department of Information Technology, Government Polytechnic, Amravati, India

[*]*Corresponding Author: harsha.vyawahare@gmail.com, Tel.: +91-95615-84815*

DOI: https://doi.org/10.26438/ijcse/v7i2.685691 | Available online at: www.ijcseonline.org

Accepted: 12/Feb/2019, Published: 28/Feb/2019

***Abstract*—** Relational Database Management System (RDBMS) has been the power house of database industry since decades. However with the evolution of big data, they are loosing their importance due to many reasons. With the rapid growth in internet and social network, graph databases are seen as an promising alternative to relational database for determining relationships optimally and quickly. These databases can be of great help to the companies struggling with traditional databases and want some new database that can replace these legacy databases. This paper presents a theoretical investigation and comparison of the relative usefulness between relational database (MySQL) and the graph database (Neo4j).

***Keywords*—**Big Data, Graph Database, Relational Database

I. Introduction

The relational database management system was created in the 1970s. Since then, its popularity has skyrocketed, and it has become a primary data storage structure in both academic and commercial pursuits for more than thirty years. It has proved to be powerful platform for business applications and has spawned, in the form of SQL, a standard language for querying databases. Although there have been different approaches over the years such as object databases or XML stores, however these technologies have never gained the same adoption and market share as RDBMS.

Way back people used database just for storing tabular data like purchase reports and finance records & relational databases were perfect for this. However, the user requirements and hardware characteristics have evolved from that time. Also the web 2.0(and now web 3.0) came by many new applications that depend on storing and processing big amount of data and it needs high availability and scalability which added more challenges to the RDB. So as digitization has progressed, spitting up ever more kinds of data, demand for alternatives to the relational model has grown.

So growing number of companies have adopted various types

of non-relational databases, commonly referred to as NoSQL databases. NoSQL databases are those databases that are non-relational, open source, distributed in nature, having high performance and horizontally scalable. Typically (with some variations) NoSQL systems are classified, depending on the data model, into the following classes.

Key-value store: In these databases the stored data is represented by a pair of key and value per record, where each key is unique and it allows accessing record's information, represented as value i.e it is a system that stores values indexed for retrieval by keys .This structure is also known as hash table where data retrieval is usually performed by using key to access value. Some of the popular key-value databases are Riak, Redis, Amazon DynamoDB (not open-source), Project Voldemort, Couchbase and Voldemort (LinkedIn)[1][2].

Document Store: These databases are designed to manage data stored in documents that use different format standards, such as, XML or JSON. These databases store and organize data as collections of documents. Some of the popular document databases we have seen are MongoDB, CouchDB, OrientDB [1][2].

Column Family: The database structure of this NoSQL type is similar to the standard RDMS since all the data is stored as sets of columns and rows. Examples: Bigtable (Google); Hypertable; Cassandra (Facebook)[1][2].

Graph Database: These databases are mostly used when the stored data may be represented as a graph with interlinked elements such as, social networking, road maps or transport routes. Examples: Neo4j; InfoGrid; Sones GraphDB; AllegroGraph; InfiniteGraph[1][2]. Of all NoSql databases graph databases are optimized for networks (social networking and website link structure), as graph is a natural way of storing connections between users.

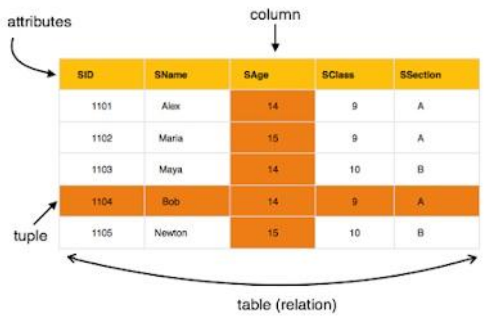
The paper is organized as follows in following subsections, Section I describes the introductory scenario behind the evolution of graph databases. Section II contains the background details of relational and graph databases including essential basics. Section III presents the related work done by various researchers in comparing relational and graph databases. Section IV describes the comparative parameters.

II. BACKGROUND

A. Relational Database

In 1970, Dr. E.F. Codd a researcher at IBM published "A Relational Model of Data for Large Shared Data Banks," an article that outlined a model for storing and manipulating data using tables. The model is based on set theory and predicate logic. This model was in contrast to the more traditional database theories of that time which were much more complicated, less flexible and dependent on the physical storage methods of the data.

A relational database at its simplest is a set of tables used for storing data. Each table has a unique name and may relate to one or more other tables in the database through common values. A table (also known as entities or relations) in a database is a collection of rows and columns. A row (records or tuples) contains data pertaining to a single item or record in a table. A column (fields or attributes) contains data representing a specific characteristic of the records in the table. A relationship is a link between two tables (i.e, relations). Relationships make it possible to find data in one table that pertains to a specific record in another table. Foreign key columns are columns that link to primary key columns in other tables, thereby creating a relationship. The standard user and application programming interface (API) of a relational database is the Structured Query Language (SQL). SQL statements are used both for interactive queries for information from a relational database and for gathering data for reports. Figure 1 shows relational database model.



SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

Figure 1. A Relational Model

Relational databases range from small, personal databases like Microsoft Access to large-scale database servers like Oracle, Microsoft SQL Server and MySQL.

B. Benefits and Drawbacks of Relational Model

Relational database has been the power house of database industry since decades. It is robust, effective, and proven technology used as one of primary means of data storage and retrieval. It is enrich with multiple positive aspects like, it avoids data redundancy by storing data at only one place due to which data inconsistency is avoided. Stick integrity rules can be enforced including foreign key constraints. It is a well tested and matured technology present in industry from decades. It has been choice of software developers due to presence of easy and convenient query language. It offers numerous security features and also employees authentication and authorization. It offers high reliability by implementing strict It carries out complex operation also. It finds it's scope in data mining, olap, oltp, business intelligence applications.

Apart from above bright side, it has some dark side also. Relational databases do not support high scalability, until a certain point better hardware can be employed but beyond that point the database must be distributed. Data is stored in relational database in form of tables, this structure gives rise to high complexity in case when data cannot be easily encapsulated in table. Relational Databases make use of SQL which is featured to work on structured data, but it can be highly complex when working with unstructured data. Recursive query support is weak. Query retrieval performance can degrade if number table in which joins have to be established increases. When the amount of data turns huge the database has to be partitioned across multiple servers, this partitioning poses several problems because joining tables in distributed servers is not an easy task. It is very rigid in schema design process, not flexible and does not capture proper meaning of data stored in it. Relational database modeling understands the strict rules for satisfying database normalization and referential integrity. It does not support dealing with unstructured data, big data, multimedia data, temporal and spatial data.

C. An Example

MySQL is a relational database system which is free to use and can downloaded from official website.. It is faster, more reliable, and cheaper than any other database system (including commercial systems such as Oracle and DB2). It is more popular with the websites. It is a light weight system which is extremely fast and is easy to use. It consist of a solid data security layer that protects sensitive data from intruders. Passwords are also encrypted. follows a client /server architecture. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, it can be increased to a theoretical limit of 8 TB of data. It is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. It allows transactions to be rolled back, commit and crash recovery.

D. Graph Database

Real world data offers a lot of possibilities to be represented as graphs thus generating undirected or directed graphs, multigraphs and hypergraphs, labeled or weighted graphs and their variants.

Graph database models can be defined as those in which data structures for the schema and instances are modeled as graphs or generalizations of them, and data manipulation is expressed by graph-oriented operations and type constructors. Technically, graph database is a way of storing data in the form of nodes and relationships. Each node represents an entity (a person, place, thing, category or other piece of data), and each relationship represents how two nodes are associated. These models took off in the eighties and early nineties alongside object oriented models. Their influence gradually died out with the emergence of other database models, in particular geographical, spatial, semistructured, and XML.

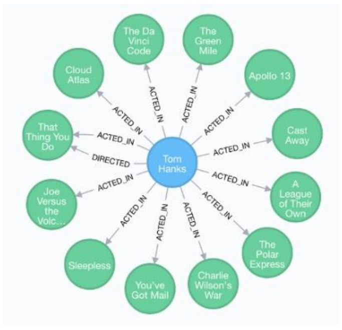


Figure 2. A simple graph database

Graph storage and graph processing are two important concepts in graph database. Certain graph databases use native or non-native storage as well as processing. Graph databases with native graph storage are optimized for graphs in every aspect, ensuring that data is stored efficiently by writing nodes and relationships close to each other. Graph storage in non native way can be done in many different ways. Some graphs can be represented as JSON or XML structures and processed by their native database tools like columnar, relational or a nosql store. These databases use other algorithms to store data about nodes and relationships, which may end up being placed far apart. This non-native approach can lead to latent results as their storage layer is not optimized for graphs. Native graph processing also called as index-free adjacency is the most efficient means of processing data in a graph because connected nodes physically point to each other in the database. However, non-native graph processing engines use other means to process Create, Read, Update or Delete (CRUD) operations.

Graph database technology contains some technological features inherent to traditional databases, e.g. ACID properties and availability. There are three generic use cases for graphs (or, indeed, any other database system): CRUD (create, read, update, delete) applications that are focused on transaction processing; query processing—reporting, business intelligence and real-time analytics (deep analytics typically in batch mode) or data discovery. Different vendors in the graph market focus on one or more of these.

E. Benefits and Drawbacks of Graph Database

Graph databases are seen as one of alternative to relational database. The main reason behind is,unlike relational db they treats relationships as first class entity. Here different kinds of nodes and edges can be used in the same database to add many layers of meaning. Thus it has simpler and more natural data modeling. Speed is another big reason graph databases are gaining traction. Graph databases may commonly be thought to be used in social network, however apart from this it finds scope in many areas like telecommunication networks, biology, chemistry and internet and it offers a sustainable competitive advantage in social media analysis, mobile data analysis and intent analysis. Graph databases wonderfully manages interconnections between objects. So they are natural fit for modeling things like interconnected web links, recommendations, tags, and friend and contact relationships. They are very well suited to the irregular, complex data involved in mapping. Graph model offers lot of flexibility in adding new nodes and relationships without compromising existing network or expensively migrating the data. With data relationships at their center, graph databases are highly efficient when it comes to query performance, even for deep and complex queries. One of the underlying strength of this model is , it is naturally indexed by relationships so they are very fast in searching also. With data relationships at their center, graph databases are highly efficient when it comes to query performance, even for deep and complex queries.They are able to quickly handle complex queries involving multiple levels of related data.

But there are limitations,as well, firstly, as with any emerging technology, conventional IT departments may struggle to find the skills required to deploy a graph database. Graph databases are not as useful for operational use cases because they are not efficient at processing high volumes of transactions. More specifically, there are currently scalability problems with most graph databases and it is difficult to partition the graph.This has made it difficult for Graph DBs to scale beyond a certain size. However, some vendors are challenging in this area .It is just a data store and doesn't give business-facing user interface to query or manage relationships. Also, it does not provide advanced match and survivorship functionality or data quality capabilities. Graph databases are not optimized for large-volume analytics queries typical of data warehousing and data mining.

F. An Example

Neo4j, a product of Neo Technologies is one of the most popular, java based open source property graph database which has bindings for other languages like ruby, python, jruby, clojure and scala. It has dual license i.e. open source and commercial. It is robust, embedded and disk-based which has native storage manager completely optimized for storing graph structures for maximum performance and scalability.It is good at graph processing & solving analytical problems that relational databases struggle to solve in a flexible way. Also it is best suited for social networking, classification of biological or medical domains, fraud detection, recommendation engines etc. It can handle graphs of several billion nodes/relationships/properties on a single machine. It has powerful traversal framework for high-speed traversals in the node space. It has simple and convenient object-oriented API. It is fully transactional like a real database and supports JTA/JTS, XA, 2PC, Tx recovery, deadlock detection, etc. Does not support sharding.

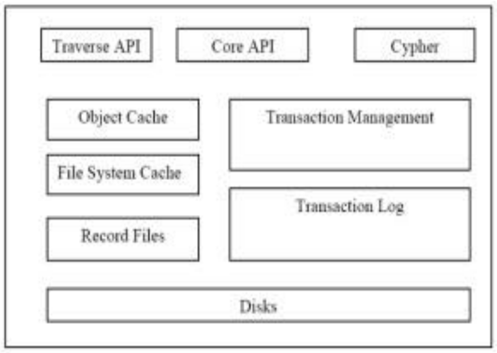


Figure 3. Neo4j Architecture

Neo4j does not require a schema, nor does it require data typing, so it is inherently very flexible. It stores graph data directly and offers large-scale horizontal scalability using replication in addition it offers ACID transactions and indexes similar to a traditional database. It also has a REST API and its own graph query language called cypher. It too supports multiple query languages like sparql and gremlin.

III. LITERATURE REVIEW

Many researchers have worked on experimental analysis of relational and graph database. Authors in [3] have evaluated MySQL and Neo4j based upon subjective (level of support/maturity, security & flexibility) as well as objective parameters. For evaluation based on objective parameters a set of predefined queries were defined by the author. In general, graph databases performed better when objective tests were performed. Thus neo4j can be used for commercial purposes like website link structures and social networking.

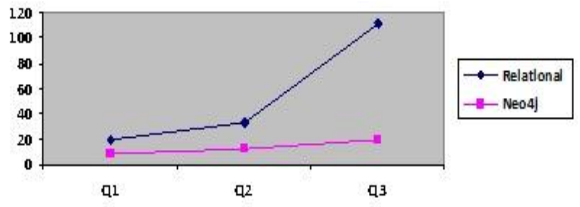


Figure 4. Retrieval times of queries by neo4j and mysql (100 objects)

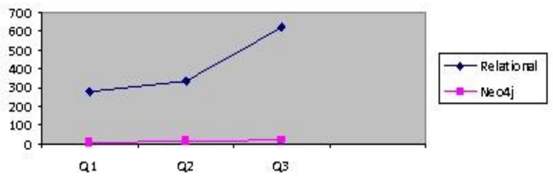


Figure 5. Retrieval times of queries by neo4j and mysql (500 objects)

Authors in [4] [5] has surveyed & compared Nosql and relational databases. Authors have focused on comparative features between nosql and relational. The main findings includes how nosql databases are different in many aspects from traditional databases like structured schema, transaction methodology, complexity, crash recovery, dealing with storing big data, features that lead to use NoSQL in cloud computing and data warehouses. NoSQL lacks in security mainly because their designer focuses on other purposes than security and generally the NoSQL databases solutions are still not matured.

Authors in [6] presents methodology designed to compare MySQL and Neo4j involving objective benchmarks and subjective comparisons. The objective tests include processing speed based on a predefined set of queries, disk space requirements, and scalability. Subjective tests include maturity/level of support, ease of programming, flexibility, and security. Both systems performed acceptably on the objective benchmark tests. In general, the graph database did better at the structural type queries than the relational database. Table 1 shows required results.

Also same kind of comparison is done by author in [7] between relational databases and graph databases for storing and processing large-scale social. Author has done two types of analysis i.e. quantitative (of storage cost and executing time) and qualitative (in terms of maturity, ease of programming, flexibility, security and data visualization). Graph database show better quantitative performance.

Table 1: Structural query results, in milliseconds

Size	Database					
	My SQL S4	*Neo4j S4*	*MySQL S128*	*Neo4j S128*	*MySQL S0*	*Neo4j S0*
1000 int	38.9	2.8	80.4	15.5	1.5	9.6
5000 int	14.3	1.4	97.3	30.5	7.4	10.6
10000int	10.5	0.5	75.5	12.5	14.8	23.5
100000int	6.8	2.4	69.8	18.0	187.1	161.8
1000 char8k	1.1	0.1	21.4	1.3	1.1	1.1
5000 char8k	1.0	0.1	21.4	1.3	1.1	1.1
10000 char8k	1.1	0.6	37.4	4.3	14.9	14.6
100000 char8k	1.1	6.5	40.9	13.5	187.1	146.8
1000 char32k	1.0	0.1	12.5	0.5	1.3	1.0
5000 char32k	2.1	0.5	29.0	1.6	7.6	7.5
10000 char32k	1.1	0.8	38.1	2.5	15.1	15.5
100000 char32k	6.8	4.4	39.8	8.1	183.4	170.0

Emil [8] talks about a very interesting experiment in a webinar on introduction to graph databases showing the performance difference between a relational database and a graph database for a problem called "arbitrary path query".Specifically, given 1,000 users with an average of 50 friend relationships each, determining if one person is connected to another in 4 or fewer hops. The results were compared against a popular open-source relational database, the query took around 2,000 ms. For a graph database, the same determination took 2 ms. So the graph database was 1,000 times faster for this particular use case. Table 2 shows the required results.

Table 2: Traversal performance comparison

Depth	RDBMS Execution time (seconds)	Neo4j Execution time (seconds)	Records Returned
2	0.016	0.01	2500
3	30.267	0.168	125,000
4	1543.505	1.359	600,000
5	Not Finished	2.132	800,000

Authors in [9] have also compared mysql (a relational db) and neo4j (a graph db) using some predefined queries. Results have shown that in storing and retrieving highly connected data, graph databases give better results. Authors in [10] also presents a same performance comparison which is not like traditional benchmarks. Author has made comparison based on twelve predefined queries for three data size configurations in the context of a real health application which was developed in Costa Rica. The results of the comparison indicate that MySQL performs better than Neo4j in most cases, but has a poor performance when data size is large and the queries have multiple join operations. Authors in [11] have also made comparison between mariaDB (a relational db) and neo4j (a graph db) and have concluded that graph databases are more scalable and flexible than relational databases as new relationships can be added to graph databases without the need to restructure the schema again. Authors have compared both on parameters like retrieval time complexity, schema load time and throughput where graph db shows better performance.

IV. Relational And Graph Database Comparative Parameters

There are various parameters on which the two databases can be compared as follows:

A. Definition

SQL/Relational databases require a structure with defined attributes to hold the data, unlike graph databases which usually allow free-flow operations. Relational databases are schema rigid while graph databases are schema free.

Relational databases work with sets while graph databases work with paths. For example when trying to emulate path operations (e.g. friends of friends), in relational database it is achieved by recursive joins. Here query latency grows unpredictably and massively as well as memory usage. While graph databases don't suffer this kind of join pain because they express relationships at a fundamental level. In graph database, the relationships are stored at the individual record level, while in a relational database, the structure is defined at a higher level i.e the table definitions. Graph databases make modelling and querying much more pleasant.

B. Transaction Reliability

Relational databases guarantee very high transaction reliability because they fully support ACID unlike the graph databases because they range from BASE to ACID.

C. Scalability

Scalability in relational databases is greatest challenge that it faces; because it depends on the vertical scalability (by adding more hardware resources like RAM, CUP, etc...) however vertical scalability dependence on improving hardware is very costive and actually impractical for the reason of hardware limitation. Other type of scalability is horizontal (in which more commodity nodes or system unites are added). NoSQL databases depend on the horizontal scalability.

D. Speed

A relational database is much faster when operating on huge numbers of records. In a graph database, each record has to be examined individually during a query in order to determine the structure of the data, while this is known ahead of time in a relational database. However graph databases are much faster than relational databases for connected data. A consequence of this is that query latency in a graph database is proportional to how much of the graph is explored in a query, and is not proportional to the amount of data stored.

E. Relationships

Despite of name, relational databases are not well-suited for today's highly connected data, because they don't robustly store relationships between data elements. Whereas graph databases treat relationships as first class citizens.
affiliations.

F. Querying

Regardless of their licences, relational databases all implement the SQL standard to a certain degree and thus, they can be queried using the Structured Query Language (SQL). NoSQL databases, on the other hand, each implement a unique way to work with the data they manage.

G. Reliability

When it comes to data reliability and safe guarantee of performed transactions, SQL databases are still the better.

H. Maturity

Relational database management systems have decade's long history. They are extremely popular and it is very easy to find both free and paid support. So they are more stable and mature. Both Oracle and MySQL have been providing extensive support for their commercial products. Relational databases have a unified language SQL. As SQL does not differ much between implementations.
Graph databases on the other hand have less market penetration and are less stable and less mature. It lacks a unified language to interact with the database as query languages supported by them (SPARQL, Gremlin and Cypher Query) differ much in implementation. Support for one implementation is not applicable to all others. Graph databases are still growing and maturing and have not undergone the same rigorous performance testing as relational databases.

I. Authors and Affiliations

Although relational databases are more mature and secure as compared to graph databases, but its schema is fixed, which makes it difficult to extend these databases and less suitable to manage ad-hoc schemas that evolve over time. Also it is difficult to chunk for distributed computing systems and sometimes it is difficult to represent the actual associations between pieces of information.

On the other hands graph databases are constantly in flux. New nodes, properties and edges are constantly added and removed as situations change – which is why one of the most prominent uses of graph databases is in social networking. This flexibility also helps out a lot when it comes to distributed computing. The malleable nature of the graph database meshes well with the cloud.

Table 3: Summary

Parameter	Relation DB	Graph DB
Definition	Data is recorded in table format with a fixed number of columns and rows.	Data is stored nodes as edges, where nodes act as entities and edges as relationships
Flexibility	Rigid Schema	Schema Free
Performance	Decreases when dealing with join-intensive query and further deteriorates as the data set gets grows.	Increases when dealing with connected data and tends to remain relatively constant, even as the data set grows
Agility	Less agile with new and big data requirements	Highly agile
Reliability	Guarantee high transaction reliability because they fully support ACID.	They range from BASE to ACID.
Application	Banking, Airlines, Universities, Telecommunications, Finance.	Social Networks, Recommendation Systems, Fraud Detection, Link Analysis, Biological Networks(protein interaction), Chemical Compounds, web graphs.
Scalability	Depends on the vertical scalability (by adding more hardware resources like RAM, CUP, etc...). limitation.	Depend on the horizontal scalability.
Relationships	Does not robustly store relationships between data elements	Treat relationships as first class citizens
Querying	all implement the SQL standard	Does not have standard query support.
Maturity	It is effective and proven technology	Not tested as relational db.

V. Conclusion and Future Scope

Graph database market is poised to blossom with innovation. At some point, 10 years from now, it will be one important player in vibrant analytics and data ecosystem. As awareness grows, adoption will follow. Once it becomes well known that graphs are easier, more convenient and faster, it will reach a tipping point with more and more people embracing them. Just because they are a niche today, does not mean that graph database will not dominate in future. Three decades ago relational databases were also confined to a few niche markets. But they have grown and eclipsed absolutely everything else that went before them. On the contrary relational database is robust, effective, and proven. It is logically consistent and easy to understand. Decades of engineering have made relational databases fast, reliable and flexible. For many kinds of data, especially data that might easily fit in a spreadsheet (demographics, inventory, sales leads), nothing beats a relational database. Thus both have it's prons, cons and application area.

REFERENCES

[1] N. Leavitt, "*Will NoSQL Databases Live Up to Their Promise?*", IEEE Transactions on Computer, ISSN: 0018-9162, Vol. **43**, No. **2**, pp. **12-14**, **2010.**

[2] V.Abramova, J.Bernardino and P. Furtado, "*Experimental evaluation of NoSQL databases*", International Journal of Database Management Systems, Vol.**6,** No**.3**, pp **1-16**, **2014**.

[3] S. Batra, C Tyagi, "*Comparative analysis of relational and graph databases*", International Journal of Soft Computing and Engineering issn: 2231-2307, Vol.**2**, No.**2**, pp **509-512**, **2012**

[4] M.A.Mohamed, O. G.Altrafi, M.O.Ismail, "*Relational vs. NoSQL Databases: A Survey*", International Journal of Computer and Information Technology (ISSN: 2279 – 0764), Vol. **3**, Iss. **3**, pp. **598-601**, **2014.**

[5] N.Jatana, S.Puri, M.Ahuja, I. Kathuria and D.Gosain, "*A Survey and Comparison of Relational and Non-Relational Database*", International Journal of Engineering Research & Technology e-issn: 2278-0181 ,Vol. **1**, No. **6**, pp **1-5**, **-**.

[6] C Vicknair, M Macias, Z Zao, Xiaofei Nan, Y. Chen and D. Wilkins, "*A comparison of a graph database and a relational database: a data provenance perspective*" , In Proceedings of the 48th Annual Southeast Regional Conference, **2010.**

[7] C Yaowen, "*Comparison of graph databases and relational databases when handling large scale social data*", Master's Thesis, College of Graduate Studies and Research, Department of Computer Science University of Saskatchewan, Saskatoon, September 2016.

[8] S. Medhi, H. K. Baruah, "*Relational Database and Graph Database: A Comparative Analysis*", Journal of Process Management – New Technologies International, Vol. **5**, No **2**, **2017**.

[9] A. Martinez, R. Mora, D. Alvarado, G. Lopez and S. Quiros, "*A Comparison between a Relational Database and a Graph Database in the context of a Personalized Cancer Treatment Application*", In Proceedings of the Alberto Mendelzon International Workshop on Foundations of Data Management at Panama **2016**.

[10] S. S. Marudkar, H. R.Vyawahare, "*Performance Analysis of Relational and Graph Database*", International Research Journal of Engineering and Technology, Vol. **5** Iss.**4**, pp **4686-4692**, **2018**

[11] A. Nayak, A. Poriya, D. Poojary, "*Type of NOSQL Databases and its Comparison with Relational Databases*" , International Journal of Applied Information Systems issn : 2249-0868 Vol.**5,** No.**4**, pp. **16-19** 2013.

[12] S. Patil, G. Vaswani, A. Bhatia , "*Graph Databases- An Overview*" International Journal of Computer Science & Information Technology, Vol. **5,** No.**1**, pp. **657-660, 2014**.

[13] R. Angles,C. Gutierrez, "*Survey of graph database models*" , ACM Computing Surveys (CSUR) , Vol. **40**, No. **1**, pp. **1–39**, **2008**.

[14] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen,and Dawn Wilkins, "*A comparison of a graph database and a relational database: a data provenance perspective*" , In Proceedings of the 48th Annual Southeast Regional Conference ACM SE , Vol.**53**, No.**1**, pp.**42:1-42:6**, **2010**.

[15] J. Partner, A. Vukotic, N. Watt, "*Neo4j in Action*", 1st ed. Mannig, **2014**.

Authors Profile

Ms. H R Vyawahare pursed Bachelor of Engineering and Master of Engineering from Sant Gadge Baba Amravati University 2011. She is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Computer Science and Engineering in Sipna College of Engineering and Technology, Amravati since 2000. She is a life member of ISTE,IETE,IE and CSI. She has published more than 15 research papers in reputed international journals including conferences. She has 19 years of teaching experience.

Dr P.P.Karde is Professor and currently heading the department of information technology in government polytechnic college Amravati. He has pursed Ph.D from Sant Gadge Baba Amravati university, Amravati. He is a member of many professional bodies including ISTE, EEE & IEEE computer society since 2013, a life member of the ISROSET since 2013 and ACM since 2011. He has published many research papers in reputed international journals including and conferences including IEEE and it's also available online. His main research work focuses on Selection &Maintenance of Materialized View.. He has 23 years of teaching experience.

Dr V.M.Thakare is Professor and Head of PG department of computer science in Sant Gadge Baba Amravati University, Amravati. He has pursued Ph.D from Dr B.A.M university, Aurangabad. He is one of senior most professor of Computer Science among all the 11 Government Universities in Maharashtra. He has guided many Ph.D scholars. He acts as expert and member in many bodies and advisory committees. He is also member of many professional bodies. He has delivered hundreds of keynote addresses and invited talks throughout India on variety of subjects related to computer science and engineering. Organized number of International and National conferences, workshops and seminars. He has published many research papers in reputed international journals including and conferences including IEEE and it's also available online. He has experience of about 30 years in teaching field. His interest of research is in Computer Architecture, Artificial Intelligence and Robotics, Database and Data warehousing & mining.