RELATIONAL DATABASE AND GRAPH DATABASE: A COMPARATIVE ANALYSIS

Surajit Medhi [1] , Hemanta K. Baruah[2]
[1]Department of Computer Science, Gauhati University, Assam, India
[2]Bodoland University, Assam, India
surajitmdh@gmail.com; hemanta_bh@yahoo.com

Original Scientific Paper
doi:10.5937/jouproman5-13553

Abstract: Since 1970, relational database models have been in use for storing, manipulating and retrieving data. The importance of relational databases is going to decrease due to the exponential growth of data as it is difficult to work with large number of joining tables. For such kinds of problems, one of the best solutions is to use graph database for storing data. The graph database can be used to store highly connected data. In this article, we are going to put forward a comparison between relational database and graph database with reference to an experiment performed.

Keywords and Phrases*: Graph, Neo4j, MySQL, cypher query language*

1. INTRODUCTION

In recent years, the importance of storing and analysing data in the form of graph has been increasing. Graph database is now used in social networks, recommendation systems, biological network, web graph etc. and these graphs are highly interconnected. In relational database, data are stored in tabular form. For less connected or static data, relational database is perfect, but for highly connected data such as the network of hyperlinks in the World Wide Web, it is complex for representing in relational database [1]. A similar kind of problem arises when we want to model the social networks like Facebook, Twitter etc. It has been accepted that the web data can also be represented and visualized using the graph database Neo4j [2]

1.1 Relational Database

Relational database is a collection of data which are stored in a tabular form the organization of which is based on the relational model proposed by E. F. Codd in 1970. In Relational database, each table contain rows and columns, rows representing records and columns representing attributes.

Let us suppose that we want to maintain a database that contains information of trains of a railway system in a particular year. The information about trains, pilots, and assignments of pilots to trains are maintained by this database. Every train has a unique number, a source city, a destination city, an arrival time and a departure time. Each pilot has a unique employee identity, a name, an address, experience in years, etc. Pilots are assigned to drive certain trains on particular days of the year. First, from the description of the database, we would identify the data objects and the relationships. Data objects are train and pilot. Relationship is assigned in the sense that different pilots can be assigned to different trains. After that we proceed to identify the attributes and the primary key for a particular data object.

So, the attributes for trains are train number, source city, destination city, arrival time and departure time. Train number is the primary key. The attributes for pilots are employee identity, name, address, experience year. Employee identity is the primary key. For the relationship “assigned to”, the attributes are the primary key of train, pilot and date.

Now we can use relational tables to represent the data object and the relationship with their attributes. The train table contains the above mentioned attributes for trains, the pilot table contains the above mentioned attributes for pilots and there is one more table for the relationship assigned to attributes like train number, employee identity and date.

1.2 Graph Database

Before going to discuss different types of graph databases, we have to know the NOSQL. NOSQL is a family of database and graph database is a part of NOSQL database. NOSQL databases are optimized for handling very large data in which elements are not closely related, and therefore expensive joins are not needed. In NOSQL large volumes of data are stored efficiently. There are different types of NOSQL databases [3]. These are: key value databases, column family stores databases, document oriented databases and graph databases. In this article, we are going to focus on graph database. Every graph database is a database which uses graph structures with nodes, edges and properties to represent and to store data. A graph model is used in graph database which allows storing the particular objects with relation between them. A graph can be expressed as G = (V, E) where V is the set of vertices (nodes) and E is the set of edges (relations). The data structure such as adjacency matrix, incidence matrix, adjacency list and linked list are used to store the graph. The Resource Description Framework (RDF) is a popular representation for graph data, and it is used as a model by some graph databases even though the primary objective of RDF was not really representation of graphs [4].

In recent years, some graph databases such as DEX, Infinite Graph, Neo4j, Orient DB, Titan etc. have been developed, which are currently used in operations for storing data over a traditional relational model. A database system Big Table has been created and used by Google [5].

1.3 Neo4j

Neo4j is an open source graph database which is implemented in Java and developed by Neo technology [11]. It uses graph structure with nodes, edges and properties to store data. It provides index free adjacency, which means that each node is a pointer. It also supports the labelled property graph model. A labelled property graph model has a number of characteristics. It contains nodes and relationships where properties and nodes can be labelled with more than one label. In this model, if we want to model a relationship among friends then we have to create different nodes for friends and connect the friends with edges.

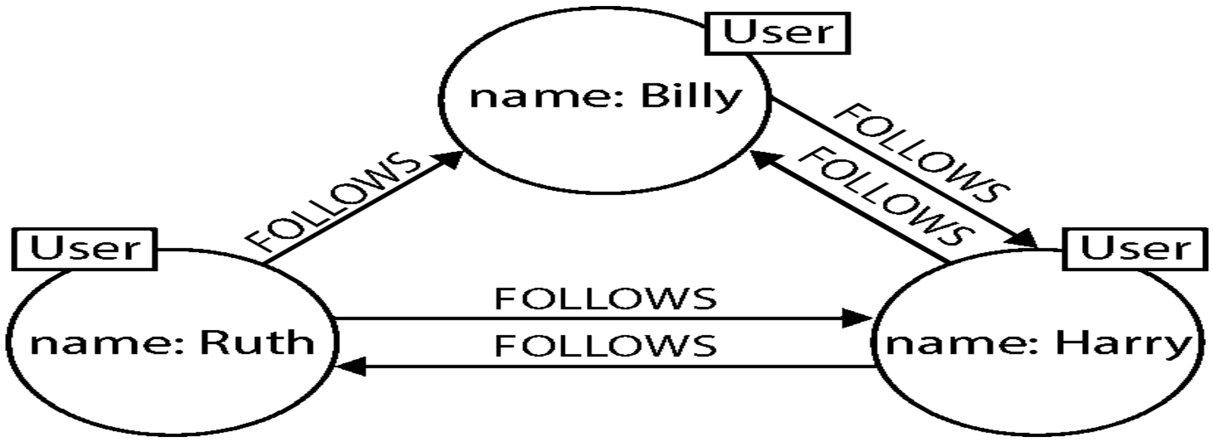

Fig. 1: Labelled Property Graph Model

In the graph model [11] shown above, there are three nodes and five relationships and a label. User is the label. Ruth, Billy and Harry are the nodes and the property is name for the nodes and 'follows' is the relationship. Here, Ruth follows Billy, which means that there is a connection between these two nodes using a directed edge, and the edge has a property called 'follows". In Neo4j, the edges are bidirectional as we can see in the model in Fig. 1. Neo4j uses the cypher query language for creating nodes, relationships with properties and also for retrieving, searching and manipulating data.

2. Related Work

Many researchers are working on graph database in different fields such as computer science, chemistry, biology, electronics etc. Balaur *et. al.* [6] have applied graph database technologies for managing comprehensive genome scale networks. In their work, they have visualized an acid metabolic network based on the cypher query language. Batra [7] also compared the relational database and graph database based on some evaluation parameters such as level of support or maturity, security and flexibility. Ching-Yung Lin [8] has worked on graph computing and linked big data, and has faced challenges to visualize the huge static graphs and has used geometry based technique to overcome this problem. Hong *et. al.* [9] used the graph database for subgraph matching with set similarity. Sakr and Al-Naymat [10] have proposed efficient relational techniques for processing graph queries. They have used two kinds of relational mapping schemes for storing graph database such as Vertex-Edge mapping and Edge-Edge mapping. In Vertex-Edge mapping, they have used two tables: Vertices (graph ID, Vertex ID, Vertex Label) and Edges (graph ID, s-vertex, d-vertex and edge label) In Edge-Edge mapping they have used only one table for representing the graph. The table contains Edge (GID, e-ID, e-label, s-VID, sv-Label, Dvid, dv-Label).

3. Relational Databases vs. Graph Databases

Relational databases were originally designed in tabular structure. Modelling relationships are not quite suitable in relational databases. It is so because they use foreign keys to relate one information with another. In relational databases, tables are needed to be joined using foreign keys to connect information from different tables. If we want to join many tables for querying, it may not be optimal to deal with.

Table 1

Employees	
Emp_id	Emp_name
E1	Raj
E2	Hirak
E3	Bimal
E4	John

Table 2

Relationships	
Emp_id	Relation_id
E1	R3
E1	R4
E2	R3
E3	R2
E3	R4
E4	R3

Here, employees and relationships are shown in two tables and the relationship table represents the relation between employees. An id has been assigned to every employee in Table 1. An emp_id in the employees' table is referenced as either emp_id or a relation_id; both the emp_id and relation_id columns in relationship table refer ids in the Employees' table. For example, Bimal has emp_id 'E3' is a relation to Raj and Hirak who has emp_id 'E1' and 'E2' respectively.

In the employees' table, emp_id is the primary key and both emp_id and relation_id are used as composite keys in the relationship table. A composite key needs both of the id's to be a unique combination of values. For example, emp_id 'E1', relation_id 'R3' and emp_id 'E1', relation_id 'R4' are not same composite value. If directed relationships are needed to be modelled, the composite value such as emp_id 'E3' and realtion_id 'R4' are completely different from the composite value than emp_id 'E4' in the relational databases. But in our example, however the relations are undirected, so that emp_id 'E3' and relation_id' R4' are the same as emp_id 'E4' and relation_id "R3".

Now, suppose we want to make simple queries such as to find out what Bimal's relationships are in the relational model. First, the query looks at the employees' table, finds which emp_id is assigned to Bimal, takes that emp_id to the relationships table, finds all relation_ids that are assigned to Bimal's emp_id, and then those relation_ids are returned back to the employees' table to find the emp_name associated to those relation_ids. This type of a query is still possible with relational model, but it is computationally expensive.

In contrast, for the same type of a model, if we use a graph database such as Neo4j then Neo4j gives us better result to model the data for above relational model. In Neo4j, the Model is the following:

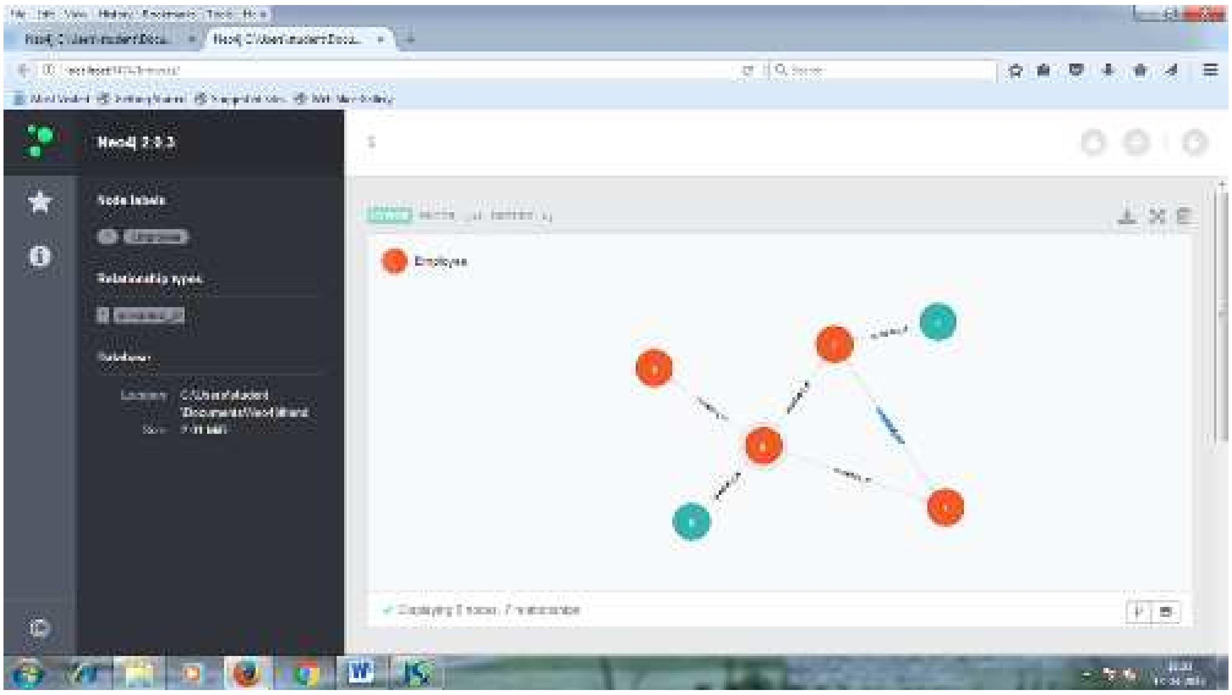

Fig. 2: Graph Model for Employee and Relationship

In Neo4j, we can easily design the model and also retrieve the information from the graph model which is computationally less expensive. Suppose, we want to make a query: “What are Bimal’s relationships?” We can easily find the result out because Bimal is connected to Raj and Hirak. So, if we compare the same query in relational model, then there would be a lot of joins and it would also be computationally expensive.

4. Implementation and Experimental Results

We have performed an experiment with a simple dataset of the game of cricket. The dataset contains the players’ information, team and which player has played Test Cricket or One Day International (ODI) matches or Twenty-Twenty (T20) matches. We would implement the cricket players’ dataset in relational database Mysql version-5.1.0 and Neo4j version-2.0.3. For Mysql PHP scripting languages are used to query the database and for Neo4j cypher query language is used to query the graph database.

The scheme for the Relational Database includes the following tables:

a) Cricketers
b) Teams and
c) Games

The Cricketers’ Table contains the attributes cric_id and cric_name. Teams’ Table contains the attributes team_id, team_name and cric_id. Games’ Table contains the attributes cric_id and game_name.

The scheme for the Graph Database includes the following nodes and relationship:

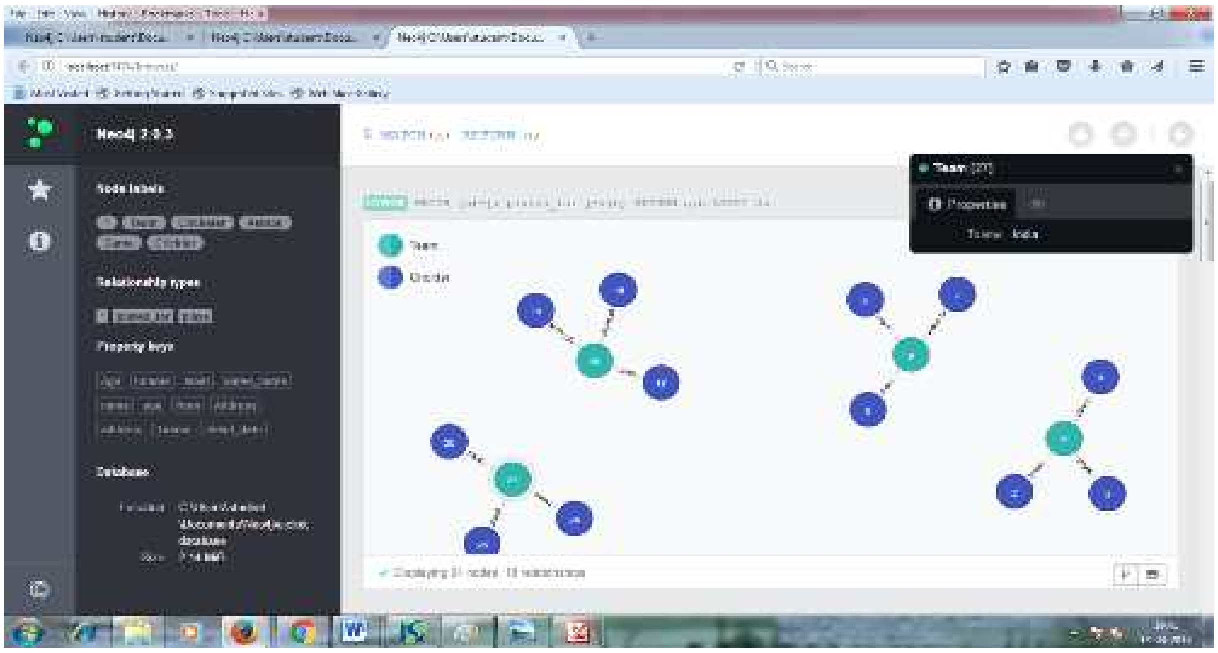

Fig. 3: Representation of Cricket Game Dataset for Relationship, Player and Team

In Fig. 3, for representing the Cricket game dataset, we have created different nodes for players, teams, game type and relationships like "belongs_to" and "plays". For example, we have created first a label "Cricketer" and then different nodes for that label such as Sachin, Dravid, Virat etc. and also set properties such as cric_id and cric_name. For team label, we have created nodes such as India, Australia, Pakistan etc. In this model, we have created the relationship between the Cricketer and the Team.

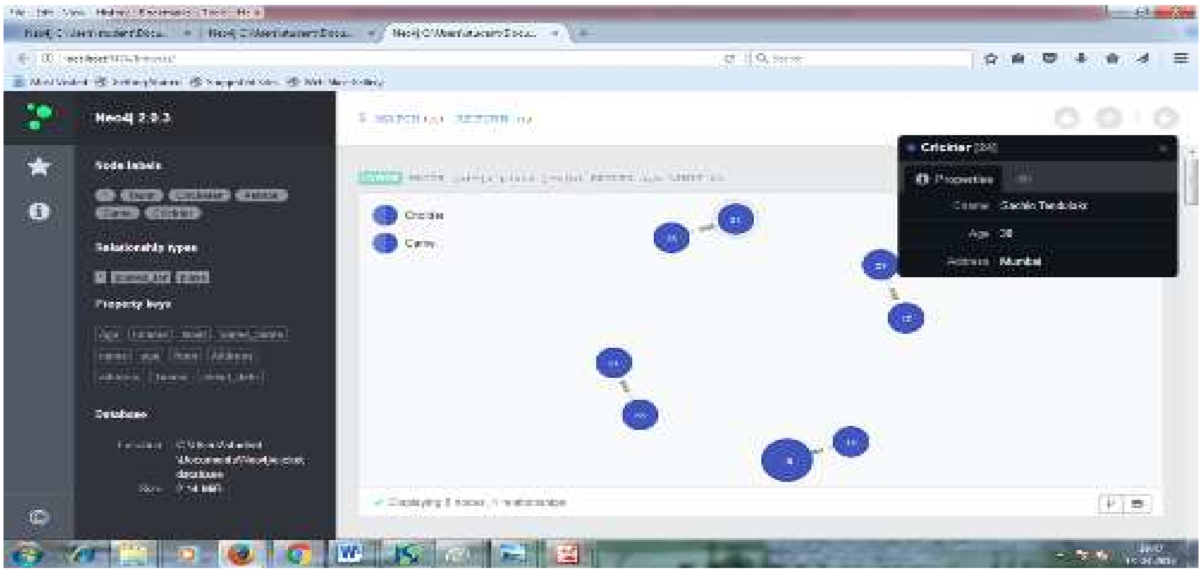

Fig. 4: Representation of the Cricket Game Dataset for Relationship, Cricketer and Game Type.

In Fig. 4, for representing the cricket game dataset with the relation between cricketer and game, we already have created different nodes for players, and we have added different nodes for game label such as Test, ODI and T20. We have set properties such as cric_id and game_name also. In this model, we have created the relationship between the cricketer and game. The objective test of evaluation including processing speed between Mysql and Neo4j is based upon a set of queries for the above schema. The Queries are as follows:

Query 1. Find the names of teams.

Query 2. Find the names of the cricketers who have played both ODI and Test Cricket.

Query 3. Find the Cricket players who belong to Team India.

In Table - 3 we have shown the query results in milliseconds (ms) for Mysql and Neo4j. We have mentioned the number of objects for the queries in Mysql and Neo4j and also have given the time taken for different queries in Mysql and Neo4j. We have calculated the time for executing the above mentioned queries for 100, 300 and 400 objects. We have drawn the column graph for the execution time for three queries in Mysql and Neo4j. It was found that graph databases are faster than relational database with respect to time taken for executing the complex queries. Further, we can design the complex relationship model easily.

Table - 3: Execution Time for Different Queries for Different Objects

No. of objects	Query1 (Mysql)	Query1 (Neo4j)	Query2 (Mysql)	Query2 (Neo4j)	Query3 (Mysql)	Query3 (Neo4j)
100	12.56 ms	5ms	18.52ms	7.32ms	15.75ms	6 ms
300	153ms	7ms	212.53ms	13ms	180.24ms	9.32ms
400	165.43ms	8.32ms	387.34ms	15.67ms	302.44ms	14.32ms

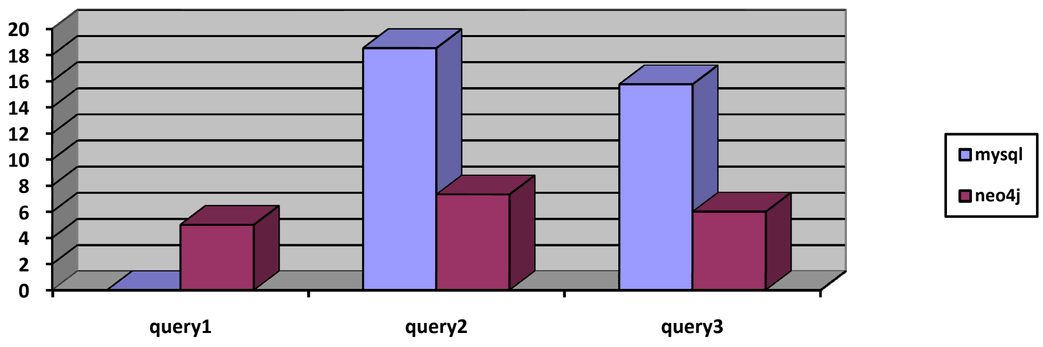

Fig. 5: Execution Time for 100 Objects in Column Graph (in Mysql and Neo4j)

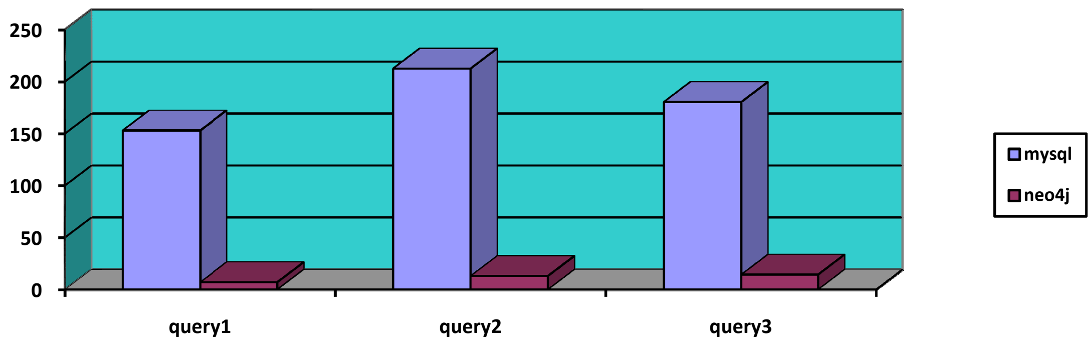

Fig. 6: Execution Time for 300 Objects in Column Graph (in Mysql and Neo4j)

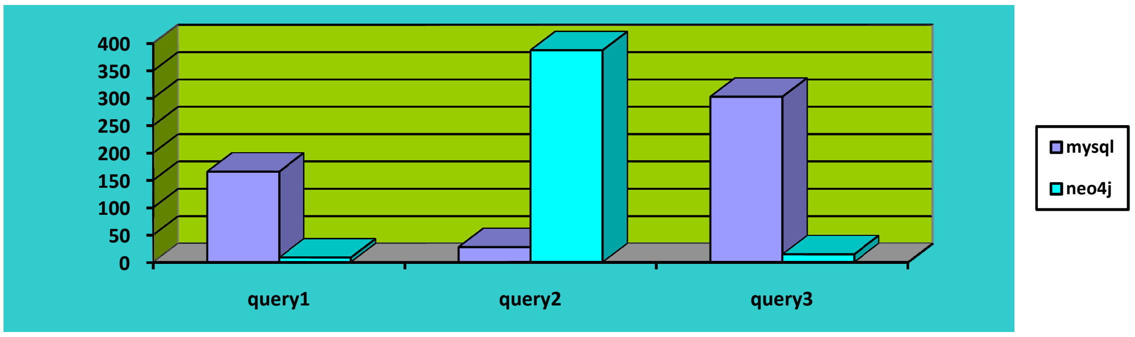

Fig. 7: Execution Time for 400 Objects in Column Graph (in Mysql and Neo4j)

5. Conclusions

Both relational database and graph database are useful to represent data. However, if we compare with reference to storing and retrieving highly connected data, graph databases appear to give better results. This means that we can more easily design and retrieve the results of queries if we use graph databases instead of using relational databases. When we want to add a new relationship to graph database, we do not need to restructure the database again. As we can see from our experiment, we get better results for executing the predefined queries in graph databases than in relational database with respect to time. The retrieval times for quires give us a conclusion that graph database is suitable to use in commercial purposes such as developing social network, stock market, recommendation engine and network management.

REFERENCES

[1] Database Trends and Applications. Available http://www.dbta.com/Articles/Columns/Notes-on-NoSQL/Graph-Dat abases-and-the-Value-They-Provide-74544.aspx,2012

[2] Surajit Medhi, Visualization of Graph Models for Web Document in Neo4j, *International Journal of Energy, Information and Communications*, Vol.7, Issue 6, 2016

[3] Gabriele Pozzani, Introduction to NoSQL, profs.sci.univr.it/~pozzani/attachments/nosql%20-%2001%20-%20introduction.pdf, April 24, 2013

[4] Jonathan Hayes, A Graph Model for RDF (Diploma thesis), Technische Universit¨at Darmstadt Universidad de Chile, 2004.

[5] R. Angles and C. Gutierrez, Survey of Graph Database Models, *ACM Comput. Surv*. 40(1):1–39, 2008.

[6] Irina Balaur, Alexander Mazein, Mansoor Saqi, Artem Lysenko, Christopher J. Rawlings and Charles Auffray, Recon2neo4j*: Applying Graph Data Based Technology for Managing Comprehensive Genome – Scale Networks*, Bioinformatics, 2017, 1–3.
.

[7] Shalini Batra and Charu Tyagi, Comparative Analysis of Relational and Graph Databases, *International Journal of Soft Computing and Engineering*, Volume-2, Issue-2, 2012

[8] Ching-Yung Lin, Graph Computing and Linked Big Data, Keynote Speech @ *International Conference on Semantic Computing*, 2014.

[9] Liang Hong, Lei Zou, Xiang Lian, Philip S. Yu, Subgraph Matching with Set Similarity in a Large Graph Database, *IEEE Transactions on Knowledge and Data Engineering*, 1041-4347, 2015.

[10] Sherif Sakr and Ghazi Al-Naymat, Efficient Relational Techniques for Processing Graph Queries, *Journal of Computer Science and Technology*, 25(6): 1237 – 1255, 2010.

[11] www.neo4j.com