Visualizing and searching relationships between academic papers using Neo4j Graph database

Thesis Report

submitted in partial fulfillment of the requirements

for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By

Karan

(801432008)

Under the supervision of:

Dr. Karamjit Kaur

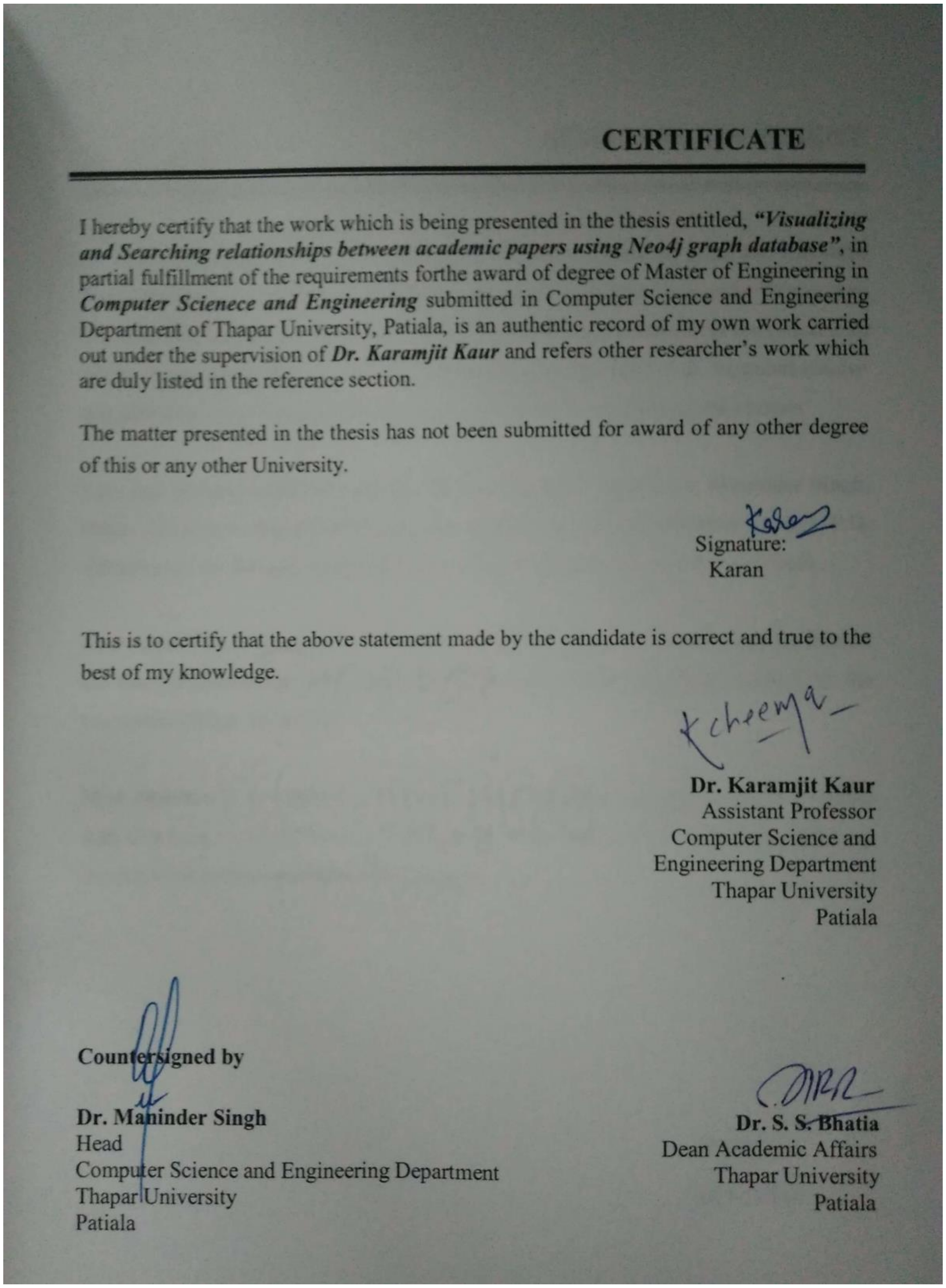
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2016



CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, ***"Visualizing and Searching relationships between academic papers using Neo4j graph database"***, in partial fulfillment of the requirements forthe award of degree of Master of Engineering in ***Computer Scienece and Engineering*** submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of ***Dr. Karamjit Kaur*** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature: [illegible]
Karan

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

[illegible]
Dr. Karamjit Kaur
Assistant Professor
Computer Science and
Engineering Department
Thapar University
Patiala

Countersigned by

[illegible]
Dr. Maninder Singh
Head
Computer Science and Engineering Department
Thapar University
Patiala

[illegible]
Dr. S. S. Bhatia
Dean Academic Affairs
Thapar University
Patiala

ACKNOWLEDGEMENT

No volume of words is enough to express my gratitude towards my guide **Dr. Karamjit Kaur**, Department of Computer Science & Engineering, Thapar University, Patiala, who has been very concerned and has aided for all the materials essentials for the preparation of this thesis report. She has helped me to explore this vast topic in an organized manner and provided me all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. S.S. Bhatia**, Dean of Academic Affairs, **Dr. Maninder Singh**, Head of Computer Science & Engineering Department and **Dr. Ashutosh Mishra**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of hour and provided with all the help and facilities, which I required, for the completion of my thesis work.

Most importantly, I would like to thank my parents and the almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Karan

(801432008)

ABSTRACT

A chasm between the obsolete Relational database management system and new generation NoSQL has led to the booming of graph database systems, which represent data as a set of nodes and relationships. But due to popularity of social networking sites s like Facebook and Twitter, where data is described by a set of nodes with relationships between them, these Graph Database systems are not in oblivion now and are reviving again with new developments. Unique visualization's and algorithms enabled by these systems are instigating more researchers towards these systems. This thesis presents software package that banks on existing Neo4j a graph database tools to provide an easily modified structure for searching and visualizing academic papers and the relationships between them. Eventually, the thesis presents a short discussion of possible future expansions of software, and shows that while representing academic papers in the graph database format shows excellent promise as a concept, much work remains in order to realize that promise.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter-1

Introduction

World Wide Web (WWW) contain large amount of data. This large amount of data is placed or stored in different or same type of databases which may have different or same schemas. The term "Information Retrieval" used when the data needed for the user is retrieved from the World Wide Web or from some kind of database. For fast and easy accessing of data Indexing may require. But due to different schemas of different databases there is difficulty in retrieving the data. So we need some kind of technique for fetching such data. There are many techniques to fetch the data but it takes huge amount of time to retrieve. There is a need of a procedure to fetch the data before user's requirement of data means prefetching the information on user prediction based on the keywords [1].

Huge data stored around the world in different type of database schemas is known as big data. Big data is combined form of structured and unstructured data. To prefetching the data in a significant way is a challenge for World Wide Web (WWW). Different types of databases are used and each database uses different schemas. To retrieve the data in an efficient way there are many different prefetching schemes.

The purpose of this thesis is to visualize the research papers using Neo4j graph database. Cypher is used as query language for storing and retrieving the data from Neo4j.The main focus of this study is to create a python tool for visualizing the research papers in Neo4j and with the help of the graph algorithm retrieve the data.

As data is increasing with a high speed there is need to handle that data there is need of databases other than relational databases. DBMS is software package which is used to handle the operations on databases like as insertion, updation and deletion. This chapter offers an introduction regarding databases. Section 1 gives an introduction of databases and evolution of databases. This section explains the factors by which non-relational databases such as NoSQL databases are so much popular over relational databases. Section 2 gives introduction to the different types of NoSQL databases and about how data is represented with various schemas. Section 3 gives introduction of Neo4j graph database and Cypher query language. Section 4 gives the structure of thesis how the thesis is organized.

1.1 Evolution of Databases

To manage the data there are different databases with different schemas. Backend work on web uses the object oriented paradigm but not all databases support this paradigm. So OODBMS comes into picture for managing the data now days. But data is increasing day by day, to handle such big data tabular schemas of relational databases become violated. Non-relational databases are popular for handling and managing big data. To handle such data scalability and reliability are the main issues. NoSQL keeps the data in one cluster, so joining and normalizing becomes easy [2].

A chasm between the obsolete Relational database management system and new generation NoSQL has led to the booming of graph database systems, which represent data as a set of nodes and relationships. But due to popularity of social networking sites s like Facebook and Twitter, where data is described by a set of nodes with relationships between them, these Graph Database systems are not in oblivion now and are reviving again with new developments. Now two popular databases, relational and non- relational databases will be discussed.

1.1.1 Relational Database Management System (RDBMS)

RDBMS is a database developed by E.F. Codd [3]. RDBMS having tabular schemas which contain arity(rows) and cardinality (columns). Before relational databases, the file based approaches are used which had various problems like as data dependencies which were overcome by RDBMS. Some categories of data integrity exist with each relational database management system like as entity integrity, domain integrity, referential integrity and user-defined integrity. Structured query language (SQL) is used as query language for most common relational databases. RDBMS has 3 layer architecture depicted in Figure 1.1 which shows the layers name and their corresponding work. Primary key is one which defines the table uniquely.

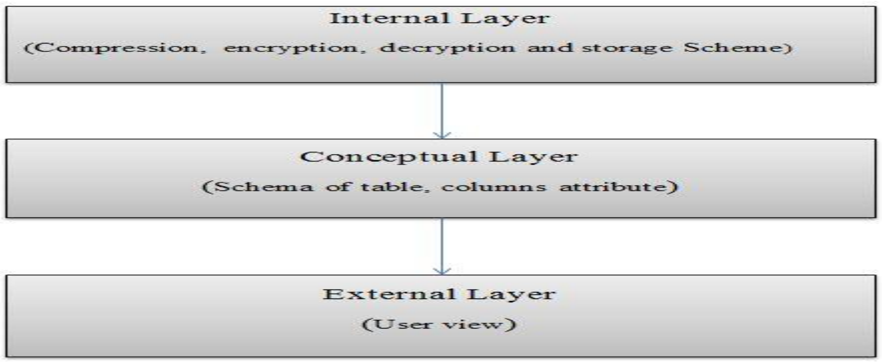

Internal Layer
(Compression, encryption, decryption and storage Scheme)
Conceptual Layer
(Schema of table, columns attribute)
External Layer
(User view)

Figure 1.1 3-layers Architecture of RDBMS

From Figure 1.1 it is clear that *Internal layer* includes the storage schema of database. It is responsible for data compression, encryption and decryption of data. *Conceptual layer* has the schema of table. It defines the fields and column attributes. It hides the details of internal layer. *External layer* is the user view.

RDMS is popular due to ACID properties [4]. Figure 1.2 explains about the ACID properties. The developers of the components that comprise the transaction are assured that these characteristics are in place.

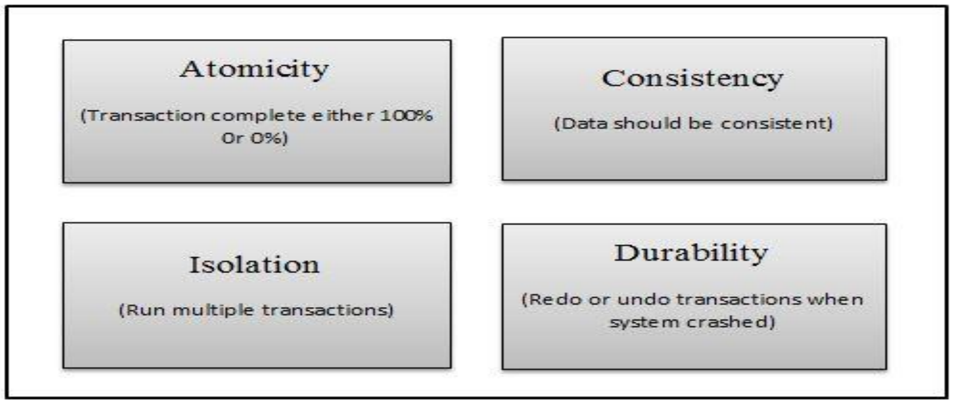

Atomicity
(Transaction complete either 100% Or 0%)
Consistency
(Data should be consistent)
Isolation
(Run multiple transactions)
Durability
(Redo or undo transactions when system crashed)

Figure 1.2 ACID properties of RDBMS

Figure 1.2 represents the core content of the ACID properties. From the figure it is clear that the atomicity asks for transaction complete to be either 100% or 0%, consistency says data should be consistent before and after each operation. Isolation means running of multiple transactions and durability means redo or undo transactions when system crashed.

1.1.2 NoSQL Foundations

NoSQL databases stands for ‘Not only SQL ‘.This term was defined by Carl Strozzi. NoSQL databases have been a new advent in this technological era to handle the growing amount of data. NoSQL databases have following three principles [5]:

- CAP Theorem
- BASE Property
- The Consistency Model

1.1.2.1 CAP Theorem:

CAP theorem is introduced by Eric Brewer in 2000 [6] and proved by Gilbert and Lynch [7]. This theorem is applied mainly on the distributed data points (store). It states about the following 3 properties with independency of data points in a distributed environment:

- Consistency: - the data should be always consistent after each operation.

- Availability: - it means the database should be always available whether there is a success or failure in operation.

- Partition Tolerance: - this property states that even there is a network failure the database nodes work fine.

In the distributed environment of the databases, CAP theorem defines the NoSQL databases in to the following 3 types of system based on the upper 3 properties which is termed as trade-offs as per CAP theorem:

- CA System
- CP System
- AP System

CAP theorem is also known as “two out of three” because it states that at any point two properties can be satisfied by NoSQL databases [8]. CA system satisfies the consistency and availability property. CP system satisfies the consistency and partition tolerance property. Neo4j is a CP system because even when there is any network failure the system remains consistent and it’s all nodes work fine. AP system satisfies the availability and partition tolerance property. However currently there are some systems that follow all three properties. Figure 1.3 shows the tradeoff of CAP theorem.

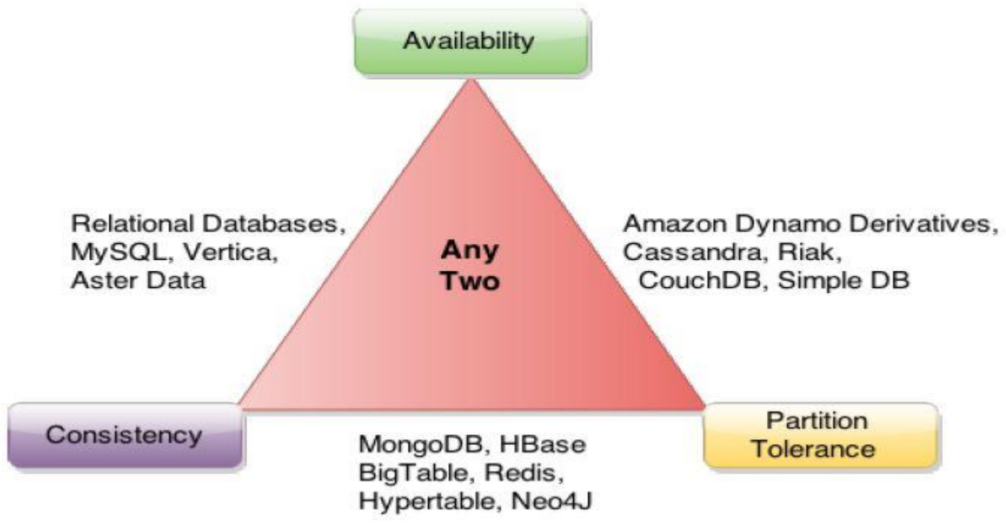


Figure 1.3 CAP theorem Trade-offs

From Figure 1.3 it is clear that a relational database system is a CP system which means it satisfies 2 properties of CAP theorem consistency and availability. RDBMS are not partition tolerance system which makes Neo4j graph database a more reliable database system. Neo4j graph database in some extent may lie in AP system.

1.1.2.2 BASE Property

NoSQL databases do not follow ACID property instead they follow BASE property which stands for: -

- Basically Available
- Soft state
- Eventually consistent

NoSQL databases follow the BASE properties which are based on the CAP theorem and works in a manner that if it has soft state and basically available data it may be eventually consistent. When data nodes are distributed, *Basically Available* property states that the data may be unavailable for a short span of time not for the whole time when there is failure or one or more nodes are down. *Soft state* property states that without given an input the state of the system can be changed which leads to dynamic designing to the developers. *Eventually consistent* property states that the system should be consistent but not necessarily after the operation. It guarantees consistency but not immediate after the operation but for some future time. So we can say that ACID properties and BASE properties are different which is shown in Table 1.1

Table 1.1 Differences between ACID and BASE properties

ACID	**BASE**
ACID stands for Atomicity, Consistency, Isolation and Durability	BASE stands for Basically Available, Soft state and Eventually consistent
RDBMS requires ACID properties	NOSQL databases require BASE properties
These properties provide reliability	These properties provide no reliability
These are complex to implement all properties	All properties can be implemented easily
The system should remain consistent after end of every operation means this guarantees consistency	The system can be in eventually consistent state

From Table 1.1 it is clear that relational databases are more reliable due to ACID properties. It is very complex task to implement ACID properties in RDBMS. It takes more cost and time if data is huge. NoSQL databases are more popular for handling the huge data. NoSQL databases take less time and cost for storing the data. Indexing of data is more difficult in RDBMS but indexing of data can be easily achieved in NoSQL databases. Navigating between ACID and BASE is totally depending on the application and application developers. Developers should select their consistency tradeoff on a case by case basis.

1.1.2.3 The Consistency Model

After the introduction of CAP theorem, the researchers done studies on consistency, availability and trade-offs between them. Consistency has different impacts on client and server side. Werner Vogel discusses about three different types of consistency models that are on the client side [9]:

- Strong Consistency
- Eventual Consistency
- Weak Consistency

The situation when write operation is done on any transaction then it returns the most recent written values; we can call the system as a *Strong Consistency* system. *Eventual consistency* situation arises when data nodes are distributed. An update in any operation will take some to reach all the nodes. *Weak consistency* does not provide guarantee of any updation means there is no guarantee after any updation in any transaction that all nodes will have the same value.

Based on the CAP theorem and the BASE properties NoSQL databases are classified. Different researchers have their different explanation on these databases. Different types of NoSQL databases are discussed in next section.

1.2 Types of NoSQL databases

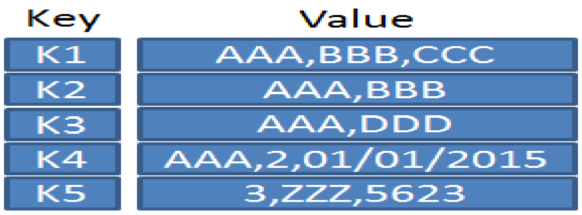
The uprising of large structured and unstructured datasets not having relational characteristics has forced large enterprises and researchers to find new and efficient ways of handling that data. Especially, NoSQL (Not only SQL) databases have become relatively dominant. In describing the NoSQL databases, Tudorica and Bucur (2011) edit the classification suggested by the NoSQL Wikipedia article (‘NoSQL’, 2014) minutely suggests that such databases can be divided into following categories [10]:

- Key-Value Data Stores
- Wide-column Data Stores

- Document-based Data Stores
- Graph-based Data Stores.

1.2.1 Key-value Data Stores

These data stores are far better than the relational databases because it stores the data in different a way. Relational databases does not allow the object-oriented concept because the structure is pre-defined for storing data in tables where as key-value databases take data as a single collection so also it takes far less memory. The data is stored simply in form of key and value. Fig 1.4 shows an example data in key value stores.



Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Figure 1.4 Data sample stored as Key-value

We can see from Figure 1.4 that keys have some id and their corresponding values. Oracle NoSQL database is a type of key-value stores. Another example includes Dynamo, couchDB etc.

1.2.2 Wide-column Data Stores

The data content is stored in each block separately column wise. Column families and Column stores are 2 types of wide-column data stores. Column families are non-relational and sparse data stores. Examples of column families are Apache HBase, Cassandra. Column stores are relational and dense data stores. Example of column stores is SQL server 2012 index.

Table 1.2 Column based Data Store

City	Pin code	Strength	Project
Noida	201301	375	25

Delhi	100006	175	15
Gurgaon	122001	350	20

Table 1.2 shows city data stores in a column data store. In Column based data store, each column stores in a separate file. So the data can be retrieved more quickly with lesser number of I/O scans. Read and write operation is done by the columns not by the rows. In column-based data stores, data Compression is more efficient since values in a column are similar to each other.

1.2.3 Document-based Data Stores

It stores the data in documents. The documents may be JSON documents or XML documents which are made up of tagged elements. Documents are ordered set of key-value pairs. Documents are gathered together in collections within the database. Document-based data stores can be used for the semi-structured data. Document-based data stores contain very powerful query engines. Document-based data stores also include indexing features. These features led to make it easy and fast to execute many different optimized queries. Some examples of document-based data stores are MongoDB, OrientDB, and Cloud-kit. Below example shows the dataset of a city stored as a document.

```
{Name: “Sector 4 Gurgaon”,
{Street: “D-31, City: “Gurgaon”, State: “Haryana”, Pincode:”122001”}}
{Latitude:”45.748328”, Longitude:”-87.985560”}}
```

In document-based data stores, there is no need of pre-structured table to handle data and to work with data. Metadata is associated with stored document in this type of database. As we have the large dataset of research papers, document based stores are not so useful to find relationships between papers as data is stored in the form of documents. Document based stores are best suited for the content oriented applications. Document based stores relies on the internal structure in the document in order to extract Meta data.

1.2.4 Graph-based Data Stores

Graph based database uses a network of nodes and edges. The network is stored in database in graph form. Complex networks easily handled by graph databases as it reduces the cumbersome of irrelevant results. Example is Neo4j graph database.

Graph Database systems have passed through multiple phases of development since its beginning 25 years ago. While the relational databases remain the most used databases for enterprise data storage, the graph data structures represent both traditional & emerging data and problem domains very well. In the current database market, particularly in enterprise data applications, relational database system remains the prevailing market leaders [11].

Nowadays, a big amount of research papers are produced each day and the user may have difficulties to search out the research paper in electronic shops or on a selected website so the NoSQL databases are a good choice to analyze such huge data. To visualize the relationships among the academic research papers, there is a need of graph database which handles the big data on web. So we select the Neo4j NoSQL graph database which helps in visualization of relationships among the academic papers. Cypher query language is a declarative language that enables performing reading and writing operations on the Neo4j graph database [12].

Today’s world is world of big data, there is huge data stored on different data servers. To retrieve that data efficiently an efficient prefetching technique is required. In existing system there are not so efficient methods to retrieve the data based on semantic of keywords. In this thesis work, concept of retrieving the academic research papers is taken which is based on the keywords used by the authors and co-authors. Till now relationships is proposed on title but not on keywords used in academic papers. The research papers graph is stored in NoSQL graph database and its cypher query language to retrieve that graph. Graph Database systems are not in oblivion now and are reviving again with new developments. Unique visualization’s and algorithms enabled by social network sites are instigating more researchers towards these systems.

1.3 Neo4j Graph Database

Neo Technology Inc. developed graph database management system. Neo4j graph database is capable to handle complex queries and highly connected data. It searches through the database with some given parameters. By providing starting point and some particular pattern, neo4j graph database searches through all the connections in neighborhood which are relevance. It collects all the information from thousands of nodes and leaves the nodes which are not relevant. Neo4j graph database is an ACID-compliant transactional database having its own local processing and storage.

Neo4j graph database visualize results through relationships among nodes. Relationship defined as directed connections with some entity provided and properties. Relationships have properties like ratings, time intervals, ranks, or costs. A direction is provided by relationships with start node, end node.

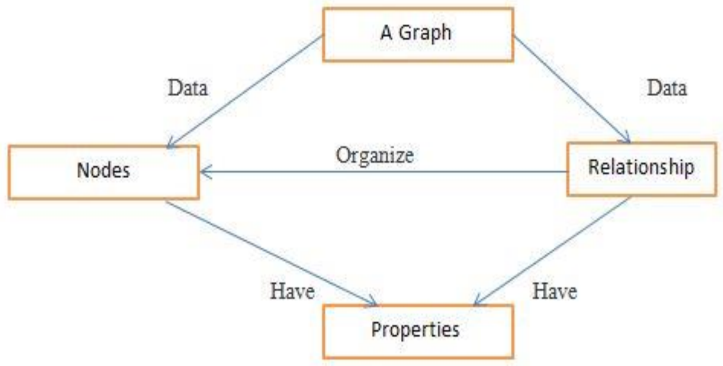


Figure 1.5 the Building Block of Property Graph

Figure 1.5 shows the building block of a graph which has some properties. Every graph consists of nodes and relationships. Data are stored in these nodes and relationships. Whenever there is a need to fetch data, graph database retrieve data according to user query. Nodes are organized by relationships through the key value. Nodes and relationship both have properties which may be common or different. JAVA is the implementation language of Neo4j graph database. Being an open source technology, Neo4j is fit for connected data transactions. Software which is written in language other than JAVA is also accessible by using Cypher Query Language.

Neo4j version 2.1.3 supports many nodes, relationships and properties to enhance the server capacity of an application. Neo4j graph database server can handle billions of nodes, relationships and labels. The latest version of Neo4j graph database server even has more capacities to handle the data in form of relationships and nodes. Table 1.3 shows the Neo4j version 2.1.3 server capacities.

Table 1.3 Neo4j Server Capacities

S.No.	Building block of Neo4j	Capacity
i.	Relationship	34 billion
ii.	Nodes	34 billion
iii.	Labels	1.75 billion

Every database requires some query language for performing different operations like as updation, insertion, deletion. Neo4j graph database uses Cypher query language to perform such functions. Next section gives a brief introduction about the Cypher query language and creation of databases in Neo4j graph database.

1.4 Cypher Query Language

Cypher query language is the declarative language used for Neo4j graph database. Cypher query language is like human language and it is human readable language. MATCH and WHERE are the main clauses of cypher query language. MATCH is used to establish the structure of pattern searched for. Extra constraints are added by WHERE clause to patterns. Suppose movie data is stored into Neo4j graph Database in the form of nodes and relationships. As an example the following cypher query will return all the movie names in which Amitabh Bachchan acted in:

MATCH (Amit: Person {name: 'Amitabh Bachchan'})-[:ACTED_IN]-(movie: Movie) RETURN movie.

In this Cypher query ‘Amit’ is a label name for Node ‘Person’. Property of Person Node is given ‘name’ and property value is given ‘Amitabh Bachchan’. ACTED_IN is a relationship that connects the Person nodes with Movie nodes. Return is a clause used for returning the graph according to user query. This query matches the nodes and relationships and returns the required graph.

1.5 Structure of Thesis

Thesis is organized in five chapters. A brief review of each chapter is as follows:

Chapter 2: This chapter includes some basic concepts the graph databases with their terminologies. A brief literature survey which I have done is given in this chapter.

Chapter 3: In this chapter, problem statement and proposed work is explained. Architecture of the proposed approach and its comparison with existing work is explained. In this chapter algorithms used are explained briefly.

Chapter 4: In this chapter, implementation of the proposed algorithm is shown using snapshots and results are discussed.

Chapter 5: This chapter concludes the thesis and the future scope of the proposed work are discussed.

In the next chapter, a brief literature review of various technologies and related work to this thesis is discussed.

Chapter-2

Literature Survey

2.1 Introduction

At present, Google and other search engines are best example of big data and semantic web. In the present era, data organization and data analytics are big issues. In recent years data increases exponentially, data can be unstructured or structured format. If data comes in structured format then we have no problem because to handle structured data we have RDBMS and many other frameworks. But if data comes in an unstructured format then we have problem because we cannot handle such a huge amount of data [13]. This section defines all the terminologies that have been used throughout this thesis.

2.1.1 Big Data

Today's world is world of Big data. In every field there is lots of data emerges and the data sets are so much large and complex that traditional methods for data processing are not enough to handle. There are many challenges to data which is stored in huge quantity and day by day it is increasing at rapid rate. Hurdles in big data like analyzing the data, searching, updating, visualization etc. make it difficult to process the data in a adequate manner. Privacy of data or information is a big challenge in Big data.

Data mining concepts along with some advanced methods of extracting the useful data are used to extract data. Predictive analytics are used to extract valuable data from big data sets. Prediction of future reference of big data may results accuracy and better decision making.

There are some important characteristics of big data:

• **Volume**. Data is stored through the years and main source of this data is transactions and queries. There are other factors also which increases the data sets stored. Social media, online shopping, online transactions and many more are the factors for heavy increase in volume of data. One machine to another machine data is collected; mirror images of data are taken. RAID systems are used to make data redundant but the storage space also increases. Because of cheaper storage memory, storing data is not a problem but retrieval of relevant data makes it difficult to manage big data.

• **Speed**. Speed plays a vital role in retrieving the relevant data from big data. There is need of quick reaction to deal with speed of big data. Data sets emerge from various fields are so huge and quick that there is need of more retrieval speed.

• **Different Formats.** Other dimension of big data is variety of data. Different organizations store different format of data which ids another hurdle in traditional data retrieval methods. Unstructured data is more complicated than traditional structured data. Data formats like email, video, audio, financial transactions, medicine datasets and other unstructured data sets require advance methods for mining and retrieval.

• **Complexity.** Complications in data comes when there are different sources are available for that data. Linking of data with other data is also a factor of complexity of data. Data mining process require more advancement and more levels of mining process to handle data. Data is linked with other data to establish relationships. Due to multiple hierarchies of relationships among data leads to loss of control of data.

2.1.2 Graph Database

A huge amount of information is produced every day and this information is linked with other information. Linking creates cumbersome relationships among information and it is very hard to find out the relevance information. Graph database is used to provide simple mechanism to link and sore the data in an efficient manner. Visualizing of relevance information in particular graph is possible through graph database system. Many research papers are produced every day and they have some keywords in common to visualize relationships among them, there is need of graph database.

Relationships are shown by a graph which contains set of nodes and nodes represent entities. Today huge amount of data is stored in different databases and that create an overload of data. Graph databases are capable to handle big data. A graph database is new way to search and visualize relevance relationships among information in an efficient way. Graph database have high degree of correlation which is advantage to many organizations [14].

2.1.2.1 Graph data modeling

Graph data models are used to establish connections among the nodes of graph database. Graph data model consists of nodes and edges which are used to make connections between nodes. The nodes and the edges show various different types of elements. Key values are defined by nodes and edges, which make connections between elements. Graph data models have preference over relational databases because complexity of data is not handled efficiently by relational data models. Complex operations are also handled by graph data models but relational data models lack some qualities. Even if dataset are small, graph data models are capable to handle it effectively. Graph data model are simple in nature even though they are very effective in complex situations.

2.1.2.2 Graph Terminology

Following scenario are used in graph database to construct the graph relationships.

Graph theory- Graph theories provides a basic structure to construct graph data model. Leonhard Euler proposed first graph theory which is used to solve the problem of Seven Bridges. Many other theorems have been stated and by result of this, there is a considerable change occurs. Various algorithms proposed based on these theories to incorporate with graph data models.

- **Fuzzy sets** – fuzzy logic is very old and popular term used to determine a relevant value. Relevant value lies between 0 and 1. This term was first used by Lotfi Zadah in 1965. Graph data model use fuzzy cluster to define a connectedness between the nodes through weighted graph.
- **Minimum spanning tree-** Graph database have edges with some weights and cost. To reduce the cost of graph, there is need to use minimum spanning tree. MST

connects all the nodes with minimum cost. Krusakal’s and Prim’s algorithms are used as MST.

- **The Hamiltonian Cycle** – MST do not allow having a cycle in a graph. Hamiltonian cycle is used to determine the shortest path as it allows to use cycle in graph. It is an algorithm which is applicable on graph database when there is a loop or cycle or bidirectional edges exist.
- **Cosine similarity –** similarity between two vectors can be determined by cosine similarity. The cosine is equal to dot product of two vectors divided by product of two vector lengths. Result 1 means perfectly similar, -1 means opposite and 0 means de-correlation.

2.1.3 Python Language

The selection of the Python language for use in this project was largely based on personal preference. However, it is important to recognize the way that this selection influences the design of the project. For instance, there are a number of libraries (including those used to access the initial metadata, and parse the scraped data) that are readily available for Python that make certain tasks easier. Conversely, the Neo4j system is built in Java, and this necessitates the use of an intermediary library rather than the library that the Neo4j developers provide, or that much of the community is focused on.

2.1.4 Keyword

A keyword is the most meaningful word within the anchor text of a URL. It’s sometimes a noun touching on some role of an affair or some object of an event or some object of a happening. E.g. ‘orange’ is a keyword in “orange is a fruit”. Research papers keywords are stored to index the papers so that papers can be easily retrieved.

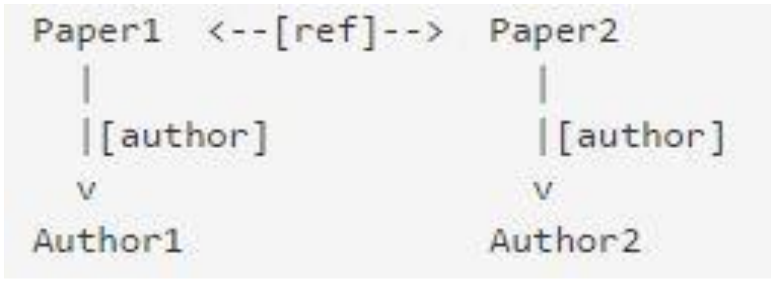
2.1.5 Ranking

Ranking tells about the no. of edges a graph has on the basis of relationships among the other nodes. Ranking of research papers is considered on the basis of relationships between papers.

2.2 Related work

Due to popularity of social networking sites where data is mapped with nodes and relationships between them, researchers are paying a lot of heed towards NoSQL databases. Recently there are many publicly released graph databases like DEX, Allegro Graph, Infinite Graph, Info Grid, but the most famous one is Neo4j Graph Database because it uses simple and powerful data model.

Rene Pickhardt (2012) in his research to get the full Neo4j power has used a data base that consists of papers and authors extracted by him from arxiv.org [11].



```
Paper1  <--[ref]-->  Paper2
  |                    |
  |[author]            |[author]
  v                    v
Author1              Author2
```

Figure 2.1 Schema of database

Figure 2.1 shows the schema of database where he tried to find coauthors which is basically a friend of a friend query following the author relationship. In his research he had concluded that newer neo4j versions the core java API became faster and cypher became slower .But in this thesis Python and cypher is used to execute friend query.

Nicholas Crouch and David M.W. Powers in their research work they implemented PyScholarGraph, a framework for searching, visualizing and indexing the academic papers [15]. They presented two ranking algorithms to find an easy way to get interesting papers within the result set. They have augmented textual search by graph based algorithms to find interesting papers but here we have used internal relationships between papers stored in Neo4j to do the same. We have modified these 2 algorithms to fit in our system which has been described in the next chapter of this thesis.

There has been a lot of work in the field of academic research. Most of the work is on researcher profiling, in which ranking the authors is the main aim. One such work is by Quinkun Zhao, which establishes the relationship between authors using community mining techniques [16]. Other such work is ArnetMiner [17] which ranks the authors on h-index and conferences on impact factor [18]. Earlier systems ranks authors on the basis of temporal locality of URL are accessed by users. Semantics governs the relationships among objects of interest for the present scenario.

Dustin Lange and Felix Naumann had proposed a model where they are matching the frequency of 2 similar words [19]. They were more concern about the duplicate words. There can be authors having same name as separate words. They are calculating the similarity measure using machine learning techniques. They used a native algorithm based on the genetic programming. Finding similarity measures in a huge data is a very difficult task but they get the best results.

Li Jun and Han Yaqing had improved the detection method of finding similarity between segments of words [20]. They proposed a paper similarity detection method that is based on the distance matrix with row-column order penalty factor. This model integrates the characteristics of vector detection, hamming distance and the longest common substring and carries out detection specific to near-synonyms, word deletion.

Konstantinos Xirogiannopoulos and Udayan Khurana had created a powerful system GraphGen for exploring the interesting graphs in relational databases [21]. They use the java library and simple queries to extract the data from DBLP dataset. Co-author graph generated by system contains schema like this : Author(ID, Name), Publication(PubID, Title, ConfID), AuthorPub(ID, PubID), PubConf(PubID, ConfID). DBLP dataset was used to analyse the power of GraphGen system. Nodes and Edges were taken as Keywords.

In the next chapter, problem statement is defined. The problems with the existing system will be discussed. The objectives of the proposed work will also be explained in the next chapter.

Chapter-3

Problem Statement

Today's world is world of big data, there is huge data stored on different data servers. To retrieve that data efficiently an efficient prefetching technique is required. In existing system there are not so efficient methods to retrieve the data based on semantic of keywords. In this thesis work, concept of retrieving the academic research papers is taken which is based on the keywords used by the authors and co-authors. Till now relationships is proposed on title not on keywords used in academic papers.

Till now, searching of research paper and finding rank of paper is based upon title only. The visualization of relationships is based upon the title only. The websites which provides the collection of research papers uses the searching technique which is based upon the title search. Broad websites like www.ieee.org, www.Springer.com etc. uses the searching technique based on title.

Researchers are interesting in most relevant results by searching. Existing system do not provide the searching and relationships among academic research papers based on the semantic of keywords. One of the disadvantages of existing work is to not giving the retrieval results on semantic of keywords. Existing system determines Relationships among research papers, movies etc. which is based upon the title of particular paper or movie. There are various keywords used in research papers and existing system do not consider relationships among them. Which leads to unnecessary results that are not relevant to what user wants the most. Visualization of relationships with ranking of keyword is not established. Time required for retrieval is more due to unnecessary results (time complexity). Cache utilization on user side is inefficient.

3.1 Objectives of the proposed work

Objectives of proposed methodology are given below:

- To find relationships among the academic research papers and movies based on the semantic of keywords used and visualize them using the Neo4j graph database. Cypher Query language is used for finding the relationships and rank of every research paper.
- To automate the prefetching technique based on relationship graph and by using the semantic of keywords of academic research papers.
- Formulate the scores for conferences and authors. According to that score the rank is calculated between papers .To find out the rank of the academic research papers based on keywords.
- To minimize the retrieval time (time complexity) by using the similarity concept.
- To use the cache memory efficiently by minimizing the miss counts.

Proposed work will be defined in the next chapter.

Chapter-4

Proposed Work

A big dataset of research papers is retrieved from the web. The database is designed according to developer point of view. There are no hard and fast rules to design a database as is the case with most of the NoSQL database systems. As there is no need to normalize the database because the dataset is already normalized so NoSQL database is used for visualizing the graph. As the relationships between research paper is to visualize so Neo4j graph database is a good choice. A good practice is to design the database which depends on the query patterns that are most used. Analysis of database should be fast. The design presented in this chapter has adequate data to perform our patterns. The proposed work is to find relationships among the academic research papers based on the semantic of keywords used and visualize them using the Neo4j graph database. Cypher Query language is used for finding the relationships and rank of every research paper. The original principle is the idea that the user could explore the graph using one paper they knew to be relevant in order to find other papers about the same topic [16]. This chapter explains about selection of language for thesis and framework of proposed work.

4.1 Selection of Language for Programming

The language used for execution of this project is python. However, it is important to consider the way that this selection affects the design of the project. For example, there are a number of libraries that are readily available for Python that make specific tasks easier.

Neo4j database is built in Java, and this initiates the use of an intermediary library rather than the library which are provided by the Neo4j developers. Py2neo is a client library in the Python language developed mostly by Nigel small that provide vendor-independent mapping to the graph domain and serves to cut short this gap [22].

In theory, the library allows establishment to any Graph Database that implements the framework, implementations for property of graph data model [11]. In practice, the implementation of graph databases can be complicated, and in fact the simplest method of using Py2neo can be to use it with version 2.3.2 of Neo4j (the current version is 3.0.1)[23]. There are so many reasons for this, most of which rotates around drastic changes made to the graph model in version 3.0 of the software (and incompatibilities with other software).

The selection of the Python language for use in this proposed work was largely based on personal preference. However, it is important to recognize the way that this selection influences the design of the project. For instance, there are a number of libraries (including those used to access the initial metadata, and parse the scraped data) that are readily available for Python that make certain tasks easier. Conversely, the Neo4j system is built in Java, and this necessitates the use of an intermediary library rather than the library that the Neo4j developers provide, or that much of the community is focused on [24].

Bulbs, or BulbFlow, is a library for the Python language developed mostly by James Thornton that aims to provide vendor-independent mapping to the graph domain (Thornton,), and serves to bridge this gap. In theory, the library allows connection to any graph database that implements the Blueprints framework [23]. Next section explains about the framework of proposed work.

4.2 Framework for Proposed work

Firstly data of research papers is collected. It can be in either of the form XML, Csv or MS Excel. The dataset is retrieved in csv format in proposed work because it is easy for python language to retrieve data in csv format. Then this dataset is stored into Neo4j graph database using py2neo module of python through which python language make the connection between python and Neo4j graph database. Neo4j graph database stores the dataset in the form of nodes and relationships. Figure 4.1 shows the framework of proposed work.

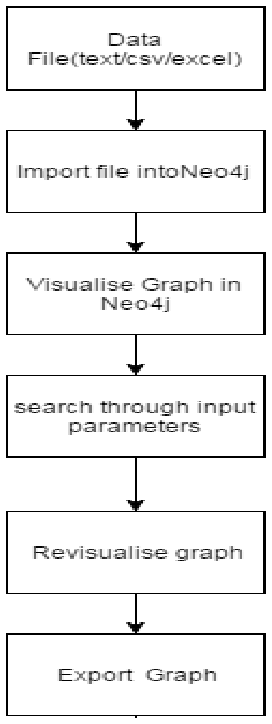


Figure 4.1 Flow diagram of Framework

Figure 4.1 explains there are various input parameters for searching or visualising the graph. The parameters used for querying the required result are as follows:

- Searching by author
- Searching by area
- Searching by similarity

Searching by author is the direct parameter and can be retrieved easily by simple cypher query and ouput will be all papers published by an author. Searching by area (in which author has published paper) is another parameter which is based on keywords of papers.One can easily find different area of an authorThe graph can be re-visualize by searching through the input parameters.

Graph can be exported into more advanced tools for better representation of results. Gephi is software used for better visualization of graph. It is an open source which runs on windows, Mac-OS-x and Linux. There are 2 ways through which we can represent out resultant graph stored in Neo4j into Gephi. Next section explains about the dataset of proposed work.

4.3 Collection of Data

A collection of dataset is taken which contain all the papers published by some faculties and students from our computer science and engineering department. The data contains name of authors, their published paper title, year of publication, paperid and keywords. The dataset is in *.csv file format. Only a cluster of records is taken to visualize the relationships among academic papers. Table 4.1 shows the sample of dataset.

Table 4.1 Dataset Created For Academic Research Papers

Author1	Author2	Ptitle	Key1	Key2	Key3	Key4
Parteek Bhatia	R. K. Sharma	UNL Based Machine …..	Machine Translation	UNL	Enconverter	Deconverter
Parteek Bhatia	Himani Jain	Hindi language interfaces….	Natural Language	Interface to Databases...	Hindi Lang....	NLP
Parteek Bhatia	Rohan Sharma	Word Sense Disambig….	word sense disambiguation	disambiguation	Hindi Language	NLP
Neeraj Kumar	Vinod Kumar Bhalla	An Efficient Multiclass….	Multiclassifier	Webpage classificatio ….	Accuracy	Internet of Thi...
Neeraj Kumar	Rajeev Tiwari	An adaptive cache ….	Cache invalidation	Cooperative caching	Cache hit ratio	Wireless…
Neeraj Kumar	Sudhanshu Tyagi	A systematic review….	Clustering	Routing	WSN	Energy …
Inderveer Chana	Nidhi Jain Kansal	Energy-aware Virtual Machine ….	Cloud computing	Energy awareness	Firefly optimization	Virtualization
Inderveer Chana	Rajni Aron	Formal QoS Policy …	Resource provisioning	Quality of service	Grid computing	Frame work
Inderveer Chana	Rajni Aron	QoS based resource…	Grid computing	Quality of service	Resource provisioning	Resource …
Inderveer	Rajni	Bacterial	Grid computing	Resource	Hyper-	Bacteri

Chana	Aron	foraging…..		scheduling	heuristic	a…
Inderveer Chana	Pankaj Deep Kaur	A resource elasticity framework ….	Cloud computing	Application behavior	Resource elasticity	Qualit y of ..
Anil Kumar Verma	Gurpreet Singh	ANTALG: An Innovative ACO…..	MANETs	Ant colony optimization	Routing	Phero mone table
Anil Kumar Verma	Virender Ranga	Network Partitioning Recovery ….	WSANs	Optimizatio n criteria	DARA	LeDir
Deepak Garg	Manoj Manuja	Intelligent text classification system …	Ontology	support vector machine	resource descriptio n ….	text classifi cat…
Inderveer Chana	Rajni Aron	Formal QoS Policy Based Grid …..	Resource provisioning	Quality of service	Grid computin g	Frame work

Table 4.1 shows the sample of Dataset to be used in this thesis. There are taken 2 author names for one paper title. 4 keywords are taken to visualize the graph of research paper according to search pattern and ranking between papers is also calculated based on these keywords. As the data is collected the database schema can be designed. Next section explains about the database schema of the proposed work.

4.4 Database Schema of proposed work

Neo4j graph database creates a relationship graph with nodes value. There must be some schema by which nodes and relationships will be stored in Neo4j graph database. As there are 2 authors in dataset having 1 paper title so there should be a relationship between the authors which contains the same paper title. Then every paper title contains the different keywords. So there should be a relationship between the paper title and different keys. There are 7 different types of node labels which will be used for the 7 different columns stored in dataset. Figure 4.2 shows the database schema of proposed work.

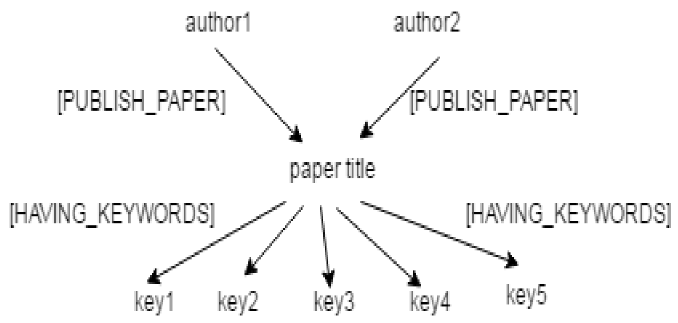


Figure 4.2 Database Schema of Proposed work

From Figure 4.2 we can see that there are 2 authors which have a relationship name to the paper title. Implementation of nodes and their relationships is achieved by the Cypher Query language and python which will be explained in the next chapter. All dataset will be stored into the Neo4j for visualizing and ranking of research papers. Next section explains about how the ranking of research papers will be achieved by Cypher query language.

4.5 Ranking of research papers

The ranking of research papers is done by finding the common keywords that papers contain. As there are 4 keywords used for ranking of keywords so a default factor of 0.25 is initially taken. This factor is named as damping factor. If the keywords taken were 5 then the dumping factor will be taken as 0.20. So we can rank the papers by the following formula:

$$\text{Rank between 2 papers} = (\text{damping factor}) * (\text{no. of common keywords})$$

A paper is compared with all the other papers in the dataset and if number of common keyword is 0 with another paper then the relationship is not built between them. Implementation part of ranking of papers will be given in next chapter. Next chapter explains about the implementation and results of proposed work.

Chapter 5

Implementation and Results

This chapter focuses upon the implementation of the work proposed and the analysis of the results obtained after the implementation is completed. Divided into four sections as follows, this chapter specifies and describes the work with the help of snapshots. The various sections are: -

- Hardware and Software used
- Framework Workflow
- Creation of Proposed Database
- Results

5.1 Hardware and Software used

Software package requires hardware for its implementation. The list of hardware is listed in the Table 4.1. This table also shows the minimum requirement to implement software package of python and Neo4j.

Table 5.1 List of Hardware Used

Name of Hardware	**Size Requirement**
Processor	32 bit
Processor Clock speed	1.8 Ghz
RAM	2 Giga Byte
Hard Disk Size	10 Giga Byte

32-bit processor is the minimum requirement for the implementation of proposed work. RAM should be 2 GB or more. Minimum hard disk size is 10 GB.

Implementation of proposed work can be implemented on window 7 operating system or on Linux because Neo4j graph database is available for both. Latest version of

Neo4j cannot be used for this implementation. Neo4j shell tools which are used for connecting the python and Neo4j are not available for the latest version. Following are the list of Software used for implementation.

Table 5.2 List of Software Used

Name of Software	**Version**
Microsoft Windows	7.0
Neo4j Graph Database	2.3.3
Python	2.7
Py2neo	2.0.8
MS Excel	2007
Notepad	6.1

Neo4j version 2.3.3 is used and all the programming is done in the python language. Py2neo module is used for making a connection between Neo4j and Python. All the datasets are stored in MS-Excel.

5.2 Creation of proposed database

The dataset of research papers is stored in the csv format. To import that data into Neo4j graph database py2neo module of python is used. Py2neo module is externally installed in python. Py2neo module connects the python and Neo4j graph database. To import this module functions in python the following line of code is written in python program:

from py2neo import authenticate, Graph

As we need to authenticate the connection between the python and Neo4j graph database, we have to use the authenticate function of py2neo module. The syntax of authenticate function is as follows:

authenticate("localhost:7474", "username", "password")

Here 7474 is the default port for opening the Neo4j in the browser. The REST API of Neo4j graph database is work with the browser which needed the username and password to authenticate. The default username and password of this API is “neo4j”.

The Graph is a class of py2neo module which contains all the properties of graph like as the definitions of nodes and relationships. To make a database of Neo4j using python the following line of code is written in python program:

graphdb = Graph("http://localhost:7474/db/data/")

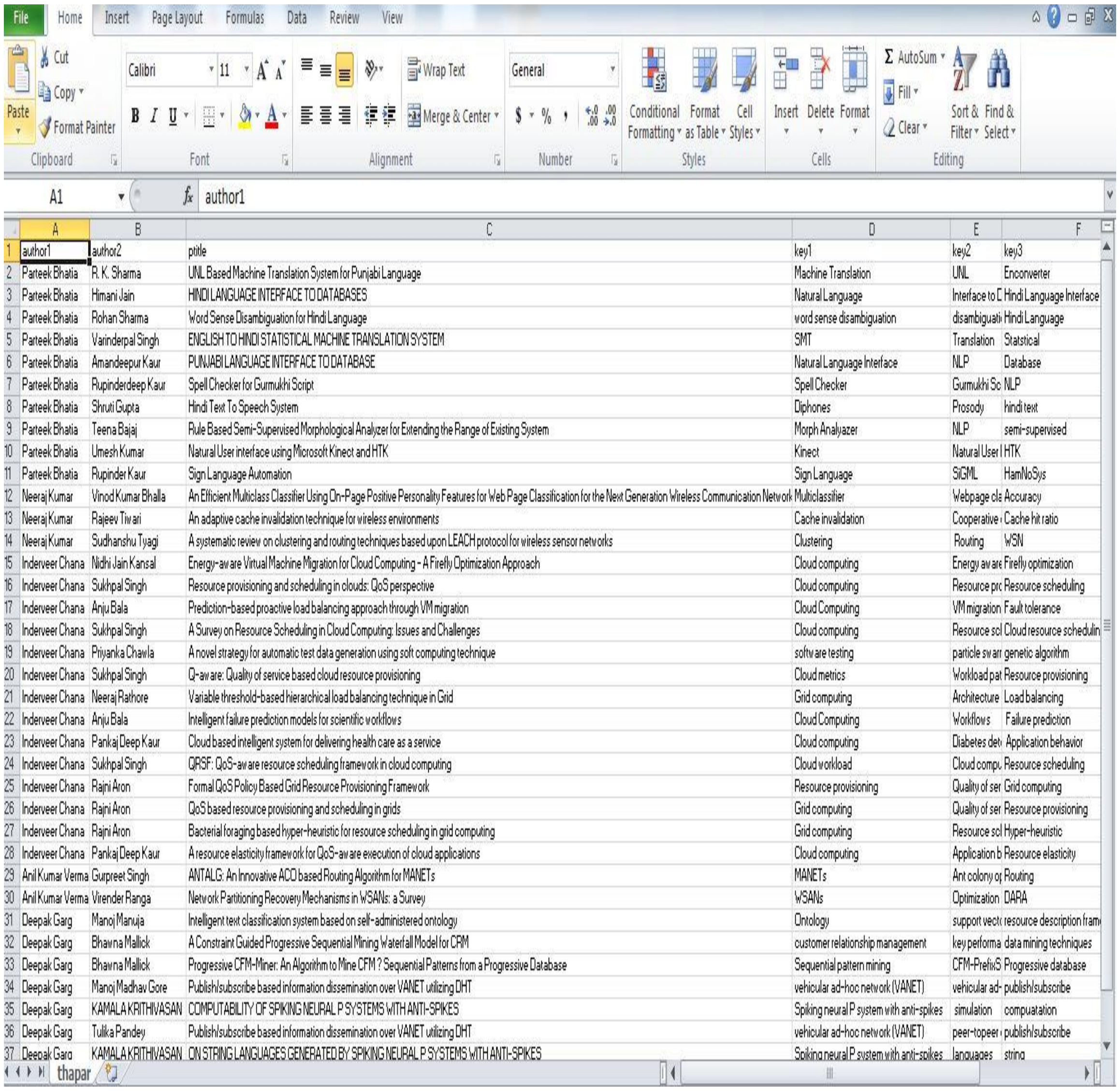
The URL written above is the default URL for generation of a graph database. Now the execution of a Cypher query can be easily achieved. Firstly the query is stored in a variable. Then following lines of code is executed for insertion into Neo4j graph database:

tx = graphdb.cypher.begin()
tx.append(INSERT_QUERY)

In step 1, tx contains the database that is enabled for Cypher query to be executed. So in the step 2, tx is appended to insert a query that is stored in the INSERT_QUERY variable. After all the query insertion for making a graph in the form of nodes and relationships the following lines of code is executed:

tx.process()
tx.commit()

In the step 1, after all insertion query python code will process all the queries on Neo4j graph database. In step 2, all transaction will save into Neo4j graph database. The dataset of research paper can be retrieved in Neo4j by 2 ways. One way is to directly load the dataset from Cypher query language, other way is to embed the Cypher query in python code and then stored into Neo4j. Because the data is in csv format, the python code itself can retrieve the data from csv file but that is a difficult task. The database of academic papers is designed in the form of nodes and relationship into Neo4j graph database. In the latest version of the Neo4j makes it possible to directly import from *.csv files in Cypher and can be used for a large dataset. The dataset taken is in *.csv file format. Figure 5.1 represents the data as a data set in which name of authors, paper title, keywords used are given.



A1 author1

	A	B	C	D	E	F
1	author1	author2	ptitle	key1	key2	key3
2	Parteek Bhatia	R. K. Sharma	UNL Based Machine Translation System for Punjabi Language	Machine Translation	UNL	Enconverter
3	Parteek Bhatia	Himani Jain	HINDI LANGUAGE INTERFACE TO DATABASES	Natural Language	Interface to D	Hindi Language Interface
4	Parteek Bhatia	Rohan Sharma	Word Sense Disambiguation for Hindi Language	word sense disambiguation	disambiguati	Hindi Language
5	Parteek Bhatia	Varinderpal Singh	ENGLISH TO HINDI STATISTICAL MACHINE TRANSLATION SYSTEM	SMT	Translation	Statstical
6	Parteek Bhatia	Amandeepur Kaur	PUNJABI LANGUAGE INTERFACE TO DATABASE	Natural Language Interface	NLP	Database
7	Parteek Bhatia	Rupinderdeep Kaur	Spell Checker for Gurmukhi Script	Spell Checker	Gurmukhi Sc	NLP
8	Parteek Bhatia	Shruti Gupta	Hindi Text To Speech System	Diphones	Prosody	hindi text
9	Parteek Bhatia	Teena Bajaj	Rule Based Semi-Supervised Morphological Analyzer for Extending the Range of Existing System	Morph Analyazer	NLP	semi-supervised
10	Parteek Bhatia	Umesh Kumar	Natural User interface using Microsoft Kinect and HTK	Kinect	Natural User I	HTK
11	Parteek Bhatia	Rupinder Kaur	Sign Language Automation	Sign Language	SiGML	HamNoSys
12	Neeraj Kumar	Vinod Kumar Bhalla	An Efficient Multiclass Classifier Using On-Page Positive Personality Features for Web Page Classification for the Next Generation Wireless Communication Network	Multiclassifier	Webpage cla	Accuracy
13	Neeraj Kumar	Rajeev Tiwari	An adaptive cache invalidation technique for wireless environments	Cache invalidation	Cooperative	Cache hit ratio
14	Neeraj Kumar	Sudhanshu Tyagi	A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks	Clustering	Routing	WSN
15	Inderveer Chana	Nidhi Jain Kansal	Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach	Cloud computing	Energy aware	Firefly optimization
16	Inderveer Chana	Sukhpal Singh	Resource provisioning and scheduling in clouds: QoS perspective	Cloud computing	Resource pro	Resource scheduling
17	Inderveer Chana	Anju Bala	Prediction-based proactive load balancing approach through VM migration	Cloud Computing	VM migration	Fault tolerance
18	Inderveer Chana	Sukhpal Singh	A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges	Cloud computing	Resource sch	Cloud resource schedulin
19	Inderveer Chana	Priyanka Chawla	A novel strategy for automatic test data generation using soft computing technique	software testing	particle swarr	genetic algorithm
20	Inderveer Chana	Sukhpal Singh	Q-aware: Quality of service based cloud resource provisioning	Cloud metrics	Workload pat	Resource provisioning
21	Inderveer Chana	Neeraj Rathore	Variable threshold-based hierarchical load balancing technique in Grid	Grid computing	Architecture	Load balancing
22	Inderveer Chana	Anju Bala	Intelligent failure prediction models for scientific workflows	Cloud Computing	Workflows	Failure prediction
23	Inderveer Chana	Pankaj Deep Kaur	Cloud based intelligent system for delivering health care as a service	Cloud computing	Diabetes det	Application behavior
24	Inderveer Chana	Sukhpal Singh	QRSF: QoS-aware resource scheduling framework in cloud computing	Cloud workload	Cloud compu	Resource scheduling
25	Inderveer Chana	Rajni Aron	Formal QoS Policy Based Grid Resource Provisioning Framework	Resource provisioning	Quality of ser	Grid computing
26	Inderveer Chana	Rajni Aron	QoS based resource provisioning and scheduling in grids	Grid computing	Quality of ser	Resource provisioning
27	Inderveer Chana	Rajni Aron	Bacterial foraging based hyper-heuristic for resource scheduling in grid computing	Grid computing	Resource sch	Hyper-heuristic
28	Inderveer Chana	Pankaj Deep Kaur	A resource elasticity framework for QoS-aware execution of cloud applications	Cloud computing	Application b	Resource elasticity
29	Anil Kumar Verma	Gurpreet Singh	ANTALG: An Innovative ACO based Routing Algorithm for MANETs	MANETs	Ant colony op	Routing
30	Anil Kumar Verma	Virender Ranga	Network Partitioning Recovery Mechanisms in WSANs: a Survey	WSANs	Optimization	DARA
31	Deepak Garg	Manoj Manuja	Intelligent text classification system based on self-administered ontology	Ontology	support vecto	resource description fram
32	Deepak Garg	Bhawna Mallick	A Constraint Guided Progressive Sequential Mining Waterfall Model for CRM	customer relationship management	key performa	data mining techniques
33	Deepak Garg	Bhawna Mallick	Progressive CFM-Miner: An Algorithm to Mine CFM ? Sequential Patterns from a Progressive Database	Sequential pattern mining	CFM-PrefixS	Progressive database
34	Deepak Garg	Manoj Madhav Gore	Publish/subscribe based information dissemination over VANET utilizing DHT	vehicular ad-hoc network (VANET)	vehicular ad-	publish/subscribe
35	Deepak Garg	KAMALA KRITHIVASAN	COMPUTABILITY OF SPIKING NEURAL P SYSTEMS WITH ANTI-SPIKES	Spiking neural P system with anti-spikes	simulation	compuatation
36	Deepak Garg	Tulika Pandey	Publish/subscribe based information dissemination over VANET utilizing DHT	vehicular ad-hoc network (VANET)	peer-topeer	publish/subscribe
37	Deepak Garg	KAMALA KRITHIVASAN	ON STRING LANGUAGES GENERATED BY SPIKING NEURAL P SYSTEMS WITH ANTI-SPIKES	Spiking neural P system with anti-spikes	languages	string

Figure 5.1 Dataset of Academic papers

Load these dataset in Neo4j graph database in the form of nodes and relationships. In new version of neo4j, import of dataset is done through integrated query. The database schema has already been defined in the previous chapter. The following are the nodes that are stored in the Neo4j graph database:

Nodes:

- Author
- Paper
- Key

Database with 200 papers by 50 authors is created.

Properties:

Author nodes will contain one property:

- Name

Paper nodes will contain following properties:

- Name
- Paperid

Key nodes will contain one property:

- Name

PUBLISH_Paper will have one property:

- Publish year

Above defined properties are used to establish relationship among nodes taken.

To execute operations such as reading and writing on graph database Cypher query langauage is a declarative langauage and used in Neo4j [17]. Cypher query language is designed to be human readable.

Here is a sample of query for creating the nodes.

Query exmaple for creating Author node:

```
MERGE (a:Author {name: 'Parteek Bhatia'})
```

This query will generate a node Author that has value name 'Parteek Bhatia'. To generate all the Author node from csv file , the cypher query is used which pick 1 coulmn at a time and then store the values of that coulmn row by row into Neo4j graph database. Similarily a cypher query can be writeen that will pick coulmns one by one and then

stores their values row by row. Sample Queries for generating paper title node and key node are given as follows.
Query exmaple for creating Paper title node:

```
MERGE(pt:Paper{ name: 'A Speech-to-Text System',year:2012})
```

Query exmaple for creating Key node:

```
MERGE (k1:key {name: 'NLP'})
```

All the keys have the same label as 'key' because all 4 columns of keys in dataset are of keywords of a paper. The relationships between the nodes are also build by the Cypher queries. Following are the relationships that are implemented in Neo4j graph database:
Relationships:
Initially 2 relationships implemented

- PUBLISH_PAPER (from Author to Paper title)
- HAVING_KEYWORDS(from Paper title to Key)

Query exmaple for creating a Relationship between author node and paper node:

```
CREATE (a)-[:PUBLISH_PAPER]->(pt)
```

Query exmaple for creating a Relationship between paper node and key node:

```
CREATE (pt)-[:HAVING_KEYWORDS]->(k1)
```

The creation of database is done when we embed all these queries into a Python code. Next section shows the result of the nodes and relationships created in the Neo4j graph database.

5.3 Results

When the python program runs on system, neo4j automatically import the dataset in graph database. Then the nodes are created for every entry in dataset. If there are redundant data in the dataset, graph database consider it different values or nodes. Graph database stores the dataset on user cache and executes it on processor. Fig.5.2 shows the nodes created for author in graph database. Cypher query used to retrieve the author node is:

```
MATCH (n:Author) RETURN n LIMIT 25
```

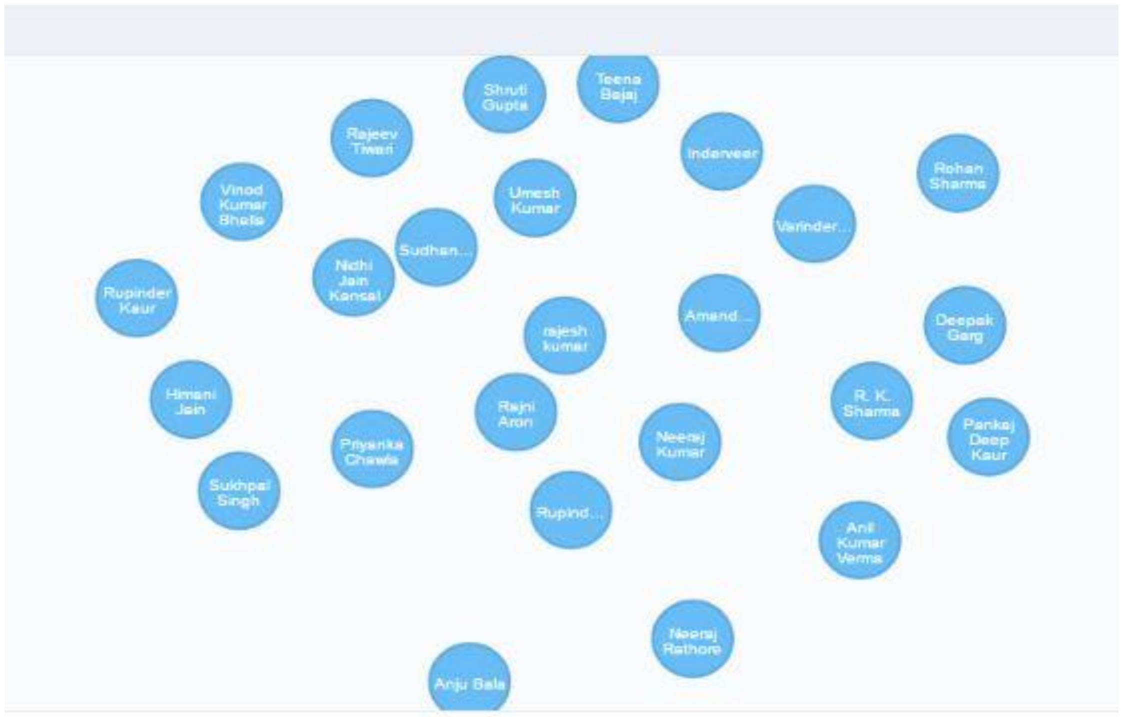


Figure 5.2 Nodes Created For Authors

Figure 5.2 shows the nodes created for authors. The query used for generating author is returning the 25 nodes. Similarly other nodes can be shown by the cypher query. Figure 5.3 shows the node created for the Key node.

Cypher query used to retrieve Key nodes:

```
MATCH (n:Key) RETURN n LIMIT 25
```

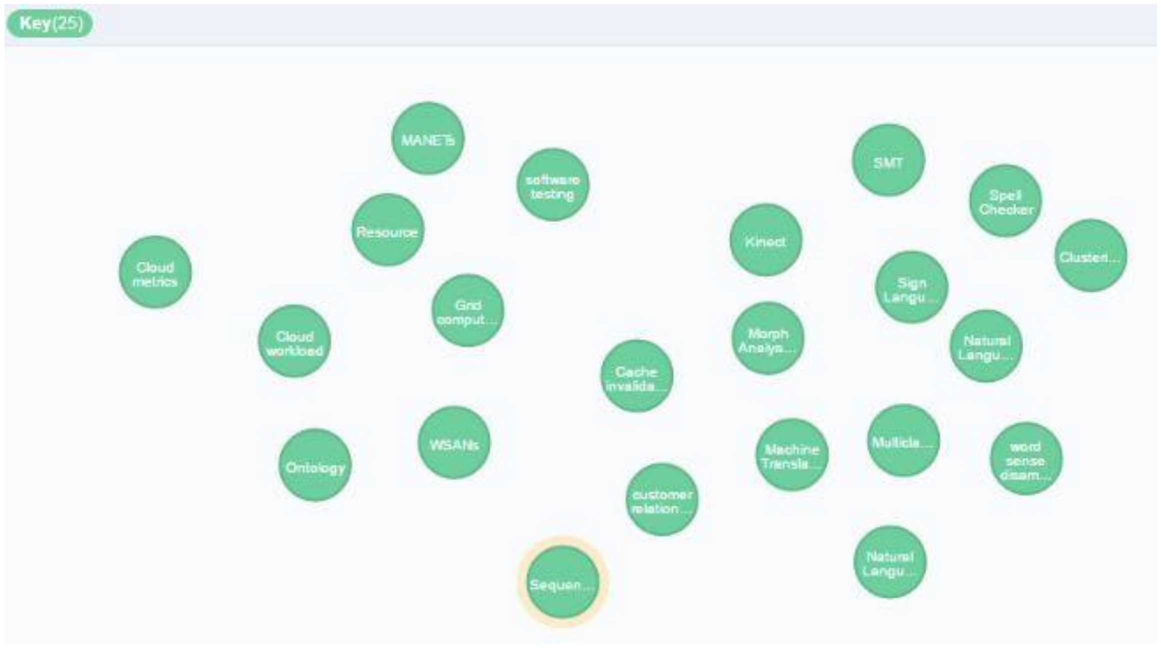


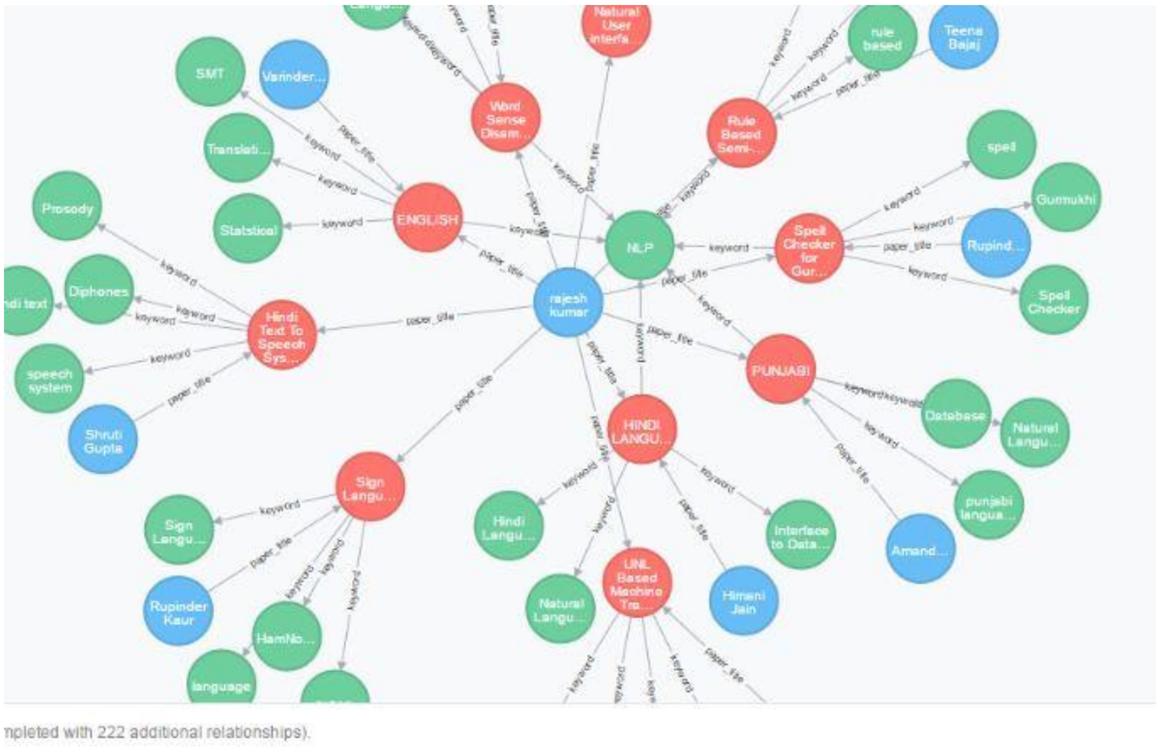
Figure 5.3 Nodes Created For Keywords

Next is to find out the relationships among all authors and among all properties of nodes. Graph database make relationships with corresponding values and the common keywords used in the dataset.

From Figure 4.2 the database schema of proposed work, there is a relationship between the authors and paper title of a research paper. Also there is a relationship between the paper title of research paper and keywords. Because now all the nodes are saved into Neo4j graph database, cypher query can be used for returning of the relationships between them. This is the term we can say visualization. Figure 5.4 shows all the relationships among all the attributes taken in dataset.

Cypher query used to retrieve is:

```
MATCH (n) RETURN (n)
```



npleted with 222 additional relationships).

Fig.5.4 Visualization of Relationship graph

The damping factor explained in the previous chapter used for the prevention of scores of research papers that falling to zero. Damping factor used to find out the rank of a research paper. Range of damping factor could be from 0.1 to 1.0. Following cypher query uses the damping factor to retrieve the rank of the research paper according to the keywords used.

```
MATCH (a:Title {name: 'QoS based resource provisioning and scheduling in grids'})—
(common)--(c:Title {name: line.ptitle})
with a, c, count(common) as in_common
MERGE (a)-[rel:RANK]->(c)
set rel.value = in_common * 0.20 return * .
```

Output of the above query is shown in Figure 5.5:

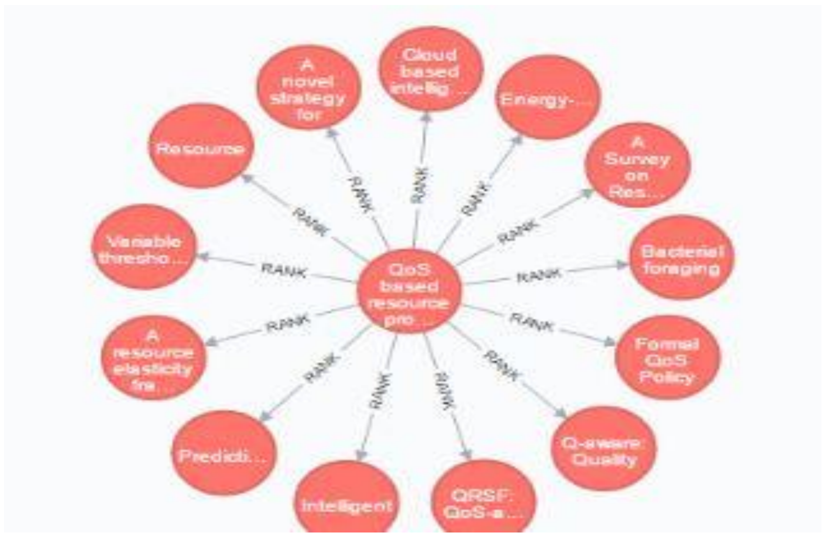


Figure 5.5 Rank relationship of a particular academic research paper

Figure 5.5 shows the rank relationship of the research paper named as 'QoS based resource provisioning and scheduling in grids' has same some keyword in other research paper also. For visualization of research papers with the common keywords, following cypher query is executed.

```
MATCH ()-[r: keyword]->() RETURN r LIMIT 25
```

The result of this query is showed in Figure 5.6.

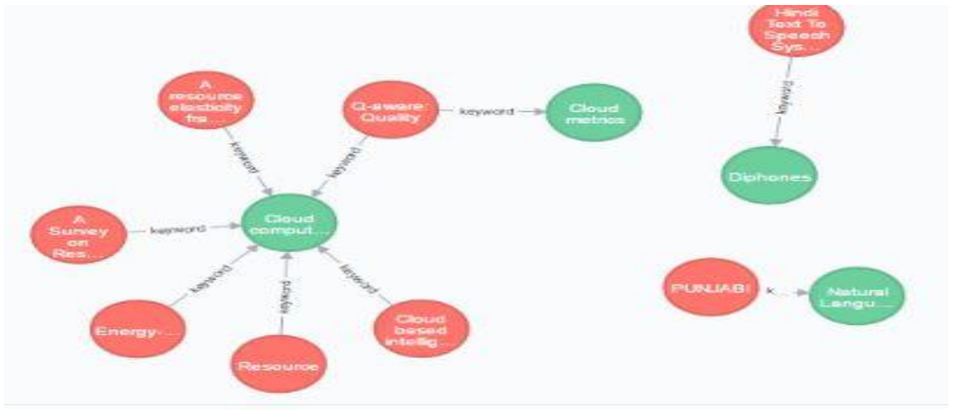
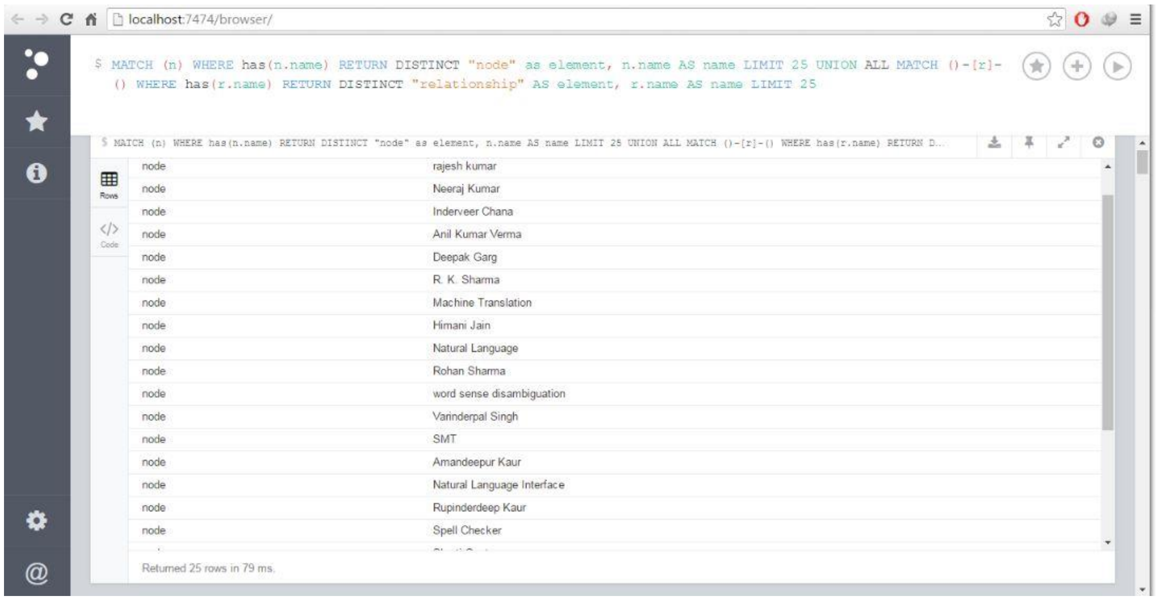


Figure 5.6 Relationship graph between Papers and Keywords

There is requirement to find out the all the research papers and the keywords which are distinct in the dataset. Then for that following query is implemented on Graph database.

MATCH (n) WHERE has(n.name) RETURN DISTINCT "node" as element, n.name AS name LIMIT 25 UNION ALL MATCH ()-[r]-() WHERE has(r.name) RETURN DISTINCT "relationship" AS element, r.name AS name LIMIT 25

Figure.5.7 shows the result of above query:



```
$ MATCH (n) WHERE has(n.name) RETURN DISTINCT "node" as element, n.name AS name LIMIT 25 UNION ALL MATCH ()-[r]-
  () WHERE has(r.name) RETURN DISTINCT "relationship" AS element, r.name AS name LIMIT 25
```

```
$ MATCH (n) WHERE has(n.name) RETURN DISTINCT "node" as element, n.name AS name LIMIT 25 UNION ALL MATCH ()-[r]-() WHERE has(r.name) RETURN D...
```

node	rajesh kumar
node	Neeraj Kumar
node	Inderveer Chana
node	Anil Kumar Verma
node	Deepak Garg
node	R. K. Sharma
node	Machine Translation
node	Himani Jain
node	Natural Language
node	Rohan Sharma
node	word sense disambiguation
node	Varinderpal Singh
node	SMT
node	Amandeepur Kaur
node	Natural Language Interface
node	Rupinderdeep Kaur
node	Spell Checker

Returned 25 rows in 79 ms.

Figure 5.7 Distinct Nodes of Research Papers and Keywords

Above result shows that running time is given by 79 milliseconds to retrieve all the distinct nodes. All the results are achieved. Figure 5.5 shows the result of a rank relationship of a paper with the other papers in dataset. The rank relationship is executed when there are common keywords between the papers. Common keywords between the papers are visualized by Cypher query as in shown Figure 5.6. Figure 5.7 shows the distinct nodes of research papers and keywords. Now this resultant graph can be export into other graph tools which will be explained in the next section.

5.4 Export Graph

Figure 5.8 shows 2 way of exporting the graph from Neo4j graph database. One way is by direct export the graph into Gephi by using the Neo4j Shell tools [29] (available on web). But there is a problem in this method because it does not work with the latest version of the Neo4j.It works only with the specific versions of the Neo4j.

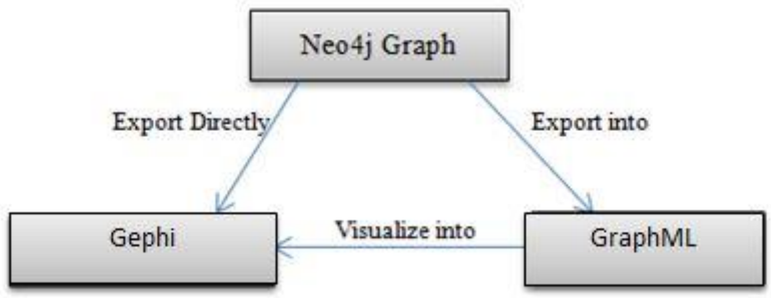


Figure 5.8 Ways of Exporting the Graph from Neo4j

Second and most common way of export the graph is exporting in the GraphML (Graph in XML format). From GraphML then the graph is export to the Gephi. This is quite large conversion of graph. Some data may be lost in between but the result set we get is more easily to understand and it shows how to play with graph. A sample graph can be shown in these modules. As there is version specific gephi and GraphML tools so there is a difficulty in extracting the graph from Neo4j Graph database.

We are finding the relationships between the various research papers based on the keywords of papers. All the nodes and relationships are stored into Neo4j graph database. One can easily re-visualize the graph according to their need. There are many possibilities of retrieving and visualizing of graph.

Next chapter will concludes what we trying to achieve so far and will also discuss the future work of the proposed work.

Chapter 6
Conclusion and Future work

In this chapter, conclusion of the research work done is presented with advantages of the proposed work. At the end of this chapter, future work is discussed.

6.1 Conclusion

The purpose of this research is to suggest a way to identify and visualize the most important research papers based on the keywords used by authors and co-authors from various fields of research. Existing method for searching is based upon title of research papers only.

In this research, a new and systematic method was discovered to achieve searching and to find out the relationships among research papers based upon keywords. Results are analyzed and a comparative study of relationships among research papers based upon keywords, titles, authors and co-authors is done.

Neo4j has some interesting features. It has used some graph algorithms like spanning tree and similarity algorithm. It can easily connect data with larger datasets. It can also easily handle and solve the problems regarding Big data analytics. There are following benefits, which proposed implementation can provide:

- Visualization of relationships among academic research papers based on keywords is efficient than existing systems.
- Execution time is reduced because integrated graph database is used.
- Works on the Big Data through NoSQL databases.
- Analyzed, visualize and compare ranking and similarity on data and relationships among research papers based upon keywords, titles, authors and co-authors.

6.2 Future Work

The above work we implemented ranking based on abstract so in future we would like to extent this rank based on conference instead of abstract. That is also could help to a person to identify his/her interesting conference. We can also extend our work by authors that would also be remarkable approach in future.

In this thesis we used a small dataset on a single node so we have the possibilities for implementing Neo4j graph database on a larger datasets. Here we are using single node system So in future we will apply on the distributed system which would take less time than single node. Neo4j works better with small datasets. Now it is going to be interesting to know how it works in a larger network.

REFERENCES

[1] Byna, Surendra, Yong Chen, and Xian-He Sun. "A taxonomy of data prefetching mechanisms." *2008 International Symposium on Parallel Architectures, Algorithms, and Networks (i-span 2008)*. IEEE, 2008.

[2] Kaur, Karamjit, and Rinkle Rani. "Modeling and querying data in NoSQL databases." *Big Data, 2013 IEEE International Conference on*. IEEE, 2013.

[3] E. F. Codd. Data models in database management. SIGMOD Rec., 11(2):112-114,june 1980.

[4] Gray, Jim. "The transaction concept: Virtues and limitations." *VLDB*. Vol. 81. 1981.

[5] Wang, Guoxi, and Jianfeng Tang. "The NoSQL principles and basic application of cassandra model." *Computer Science & Service System (CSSS), 2012 International Conference on*. IEEE, 2012.

[6] Brewer, Eric A. "Towards robust distributed systems." *PODC*. Vol. 7. 2000.

[7] Gilbert, Seth, and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." *ACM SIGACT News*33.2 (2002): 51-59.

[8] Brewer, Eric. "Pushing the cap: Strategies for consistency and availability."*Computer* 45.2 (2012): 23-29.

[9] Vogels, Werner. "Eventually consistent." *Communications of the ACM* 52.1 (2009): 40-44.

[10] Tudorica, Bogdan George, and Cristian Bucur. "A comparison between several NoSQL databases with comments and notes." *2011 RoEduNet International Conference 10th Edition: Networking in Education and Research*. IEEE, 2011.

[11] Pickhardt, Rene, et al. "Efficient graph models for retrieving top-k news feeds from ego networks." *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, 2012.

[12] Huang, Hongcheng, and Ziyu Dong. "Research on architecture and query performance based on distributed graph database neo4j." *Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on*. IEEE, 2013.

[13] Shadbolt, Nigel, Tim Berners-Lee, and Wendy Hall. "The semantic web revisited." *IEEE intelligent systems* 21.3 (2006): 96-101.

[14] Angles, Renzo, and Claudio Gutierrez. "Survey of graph database models."*ACM Computing Surveys (CSUR)* 40.1 (2008): 1.

[15] CROUCH, NICHOLAS, and DAVID MW POWERS. "PyScholarGraph: A graph-based framework for indexing, searching and visualising relationships between academic papers." *The ANU Undergraduate Research Journal*: 161.

[16] Zhao, Qiankun, et al. "Characterizing and predicting community members from evolutionary and heterogeneous networks." *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008.

[17] Tang, Jie, et al. "Arnetminer: extraction and mining of academic social networks." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008.

[18] Garfield, Eugene. "The history and meaning of the journal impact factor."*Jama* 295.1 (2006): 90-93.

[19] Lange, Dustin, Christoph Böhm, and Felix Naumann. "Extracting structured information from Wikipedia articles to populate infoboxes." *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010.

[20] Li, Jun, Yaqing Han, and Yan Niu. "A Similarity Detection Method Based on Distance Matrix Model with Row-Column Order penalty Factor." *Bulletin of Electrical Engineering and Informatics* 3.4 (2014): 285-292.

[21] Xirogiannopoulos, Konstantinos, Udayan Khurana, and Amol Deshpande. "GraphGen: exploring interesting graphs in relational data." *Proceedings of the VLDB Endowment* 8.12 (2015): 2032-2035.

[22] Han, Jing, et al. "Survey on NoSQL database." *Pervasive computing and applications (ICPCA), 2011 6th international conference on*. IEEE, 2011.

[23] ‘Neo4j, the World’s Leading Graph Database’. (n.d.). Retrieved 2016, from www.neo4j.org/.

[24] Singh, Aditya Pratap, Kumar Shubhankar, and Vikram Pudi. "An efficient algorithm for ranking research papers based on citation network." *Data Mining and Optimization (DMO), 2011 3rd Conference on*. IEEE, 2011.

List of Publications

Karan, Karamjit Kaur, “Visualizing and Searching relationships between academic papers using Neo4j graph database”, International Conference on Inventive Computation Technologies (ICICT 2016), IEEE Explore, 2016,Coimbatore, August 27,2016**[Accepted]**

Video Link

Here is the video link of the thesis:

https://www.youtube.com/channel/UC34f_A1pAFfvEHvfQDH09eQ