Comparative Analysis of Relational Databases and Graph Databases

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Charu Tyagi
(801032005)

Under the supervision of:
Dr Shalini Batra
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Comparative Analysis of Relational Databases and Graph Databases*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science 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. Shalini Batra* 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.

(Charu Tyagi)

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

(Dr. Shalini Batra)
Assistant
Professor,
Computer Science and Engineering Department,
Thapar University, Patiala

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

No volume of words is enough to express my gratitude towards my guide **Dr. Shalini Batra**, 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. Maninder Singh**, Head of Computer Science & Engineering Department and **Dr. Inderveer Channa**, 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.

Charu Tyagi

(801032005)

Abstract

Relational model has been dominating the computer industry since the 1980s mainly for storing and retrieving data. Lately, however, relational database is losing its importance due to its dependence on a rigid schema which makes it difficult to add new relationships between the objects. Another important reason of its failure is that generally, the relational model works best when there are a relatively small and static number of relationships between objects. It has long been a tricky problem in the relational databases to work with dynamic, recursive or complex relationships.

To add on to these problems, the global rapid development and maturing of Internet prompted the exploding increase of information in each field and provided people with abundant sources for seeking and obtaining useful information. This huge repository of unstructured data has resulted in making the data search and knowledge extraction, a very cumbersome task if one continues using the legacy relational databases.

One of the proposed solutions is to shift to the graph databases as they aspire to overcome such type of problems. Graph databases not only makes it easier to store and retrieve data for the World Wide Web but also provide relevant and streamlined search results from the abundant information present on the internet comparatively faster than relational databases. This thesis provides a comparative analysis of a graph database Neo4j with the most prevalent relational database MySQL.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

A database management system (DBMS) is the software that allows a computer to perform database functions of storing, retrieving, adding, deleting and modifying data. It provides facilities for controlling data access, enforcing data integrity, managing concurrency control, recovering the database after failures and restoring it from backup files, as well as maintaining database security.

Databases have been in use since many decades. The vast majority of older systems were tightly linked to the custom databases in order to gain speed at the expense of flexibility. Originally DBMSs were found only in large organizations with the computer hardware needed to support large data sets.

1.1 Database Management Systems

A database is an integrated collection of data records, files, and other database objects [1].

A database engine may comply with a combination of any of the following:

- The database is a collection of tables, files or datasets.
- Each table is a collection of fields, columns or data items.
- One or more columns in each table may be selected as the primary key.
- There may be additional unique keys or non-unique indexes to assist in data retrieval.
- Columns may be fixed length or variable length.
- Records may be fixed length or variable length.
- Table and column names may be restricted in length (8, 16 or 32 characters).
- Table and column names may be case-sensitive.

Over the years there have been several different ways of constructing databases, including:

- The Hierarchical Data Model
- The Network Data Model
- The Relational Data Model

1.1.1 THE HIERARCHIAL DATA MODEL

In hierarchical data model, data is structured in a tree of records, with each record having one parent record and many children (Figure 1.1)

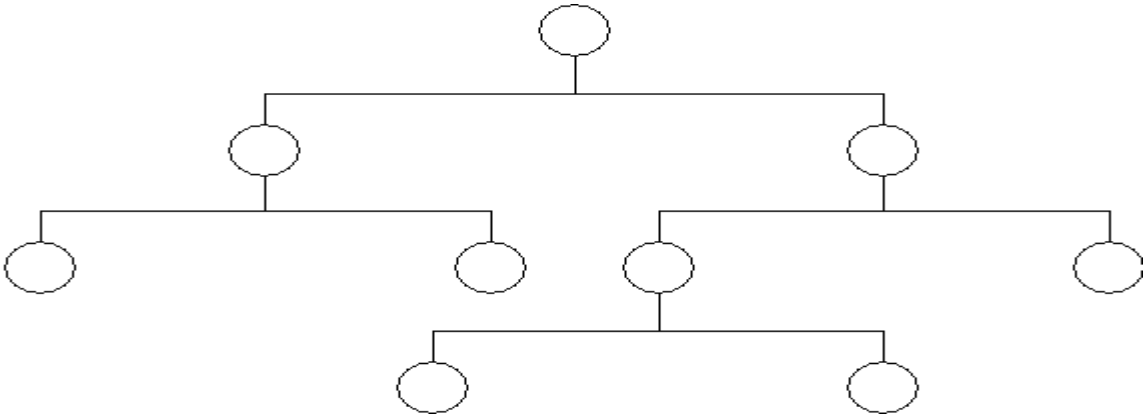


Figure1.1The Hierarchical Data Model [1]

A hierarchical database consists of:

i. Nodes connected by branches.
ii. The topmost node is called the root.
iii. Multiple nodes that appear at the top level are called root segments.
iv. The parent of node n^x is a node directly above n^x and connected to n^x by a branch.
v. Each node except the root has exactly one parent.
vi. The child of node n^x is the node directly below n^x and connected to n^x by a branch.
vii. One parent may have many children.

1.1.2 The Network Data Model

The Network Data Model uses a lattice structure in which a record can have many parents as well as many children (Figure 1.2)

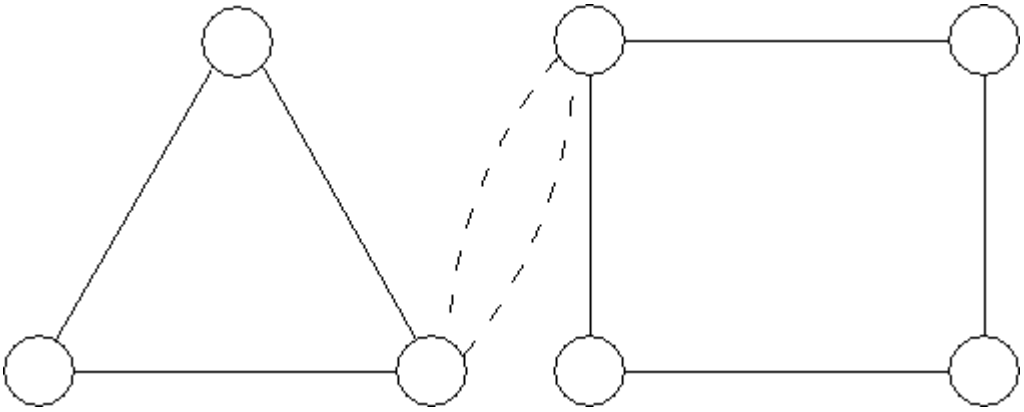


Figure 1.2 The Network Data Model [1]

Like the The Hierarchical Data Model the Network Data Model also consists of nodes and branches, but a child may have multiple parents within the network structure instead of being restricted to just one.

1.1.3 Relational DBMS

The relational database model was modeled by E. F. Codd in 1969, a researcher at IBM. The model is based on set theory and predicate logic. The basic idea behind the relational model is that a database consists of a series of unordered tables (or relations) that can be manipulated using non-procedural operations that return tables. This model was in contrast to the more traditional database theories of the time that were much more complicated, less flexible and dependent on the physical storage methods of the data.

The word relational has its roots in the terminology that Codd used to define the relational model. The table in Codd's model was actually referred to as a relation (a related set of information). In fact, Codd (and other relational database theorists) used the terms relations, attributes and tuples where most of us use the more common terms tables, columns and rows, respectively.

1.1.3.1 The Relation

The Relation is the basic element in a relational data model.

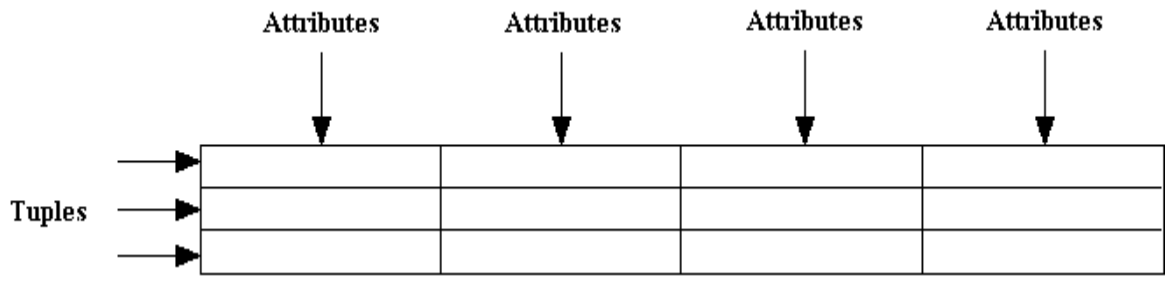


Figure 1.3 - Relations in the Relational Data Model [1]

A relation is subject to the following rules:

i. A Relation is a two-dimensional table.
ii. Attribute (i.e. field or data item) is a column in the table
iii. Each column in the table has a unique name within that table.
iv. Each column is homogeneous which means that the entries in any column are all of the same type (e.g. age, name, employee-number, etc).
v. The set of possible values that can appear in that column are called domain.
vi. A row in the table is known as the tuple.
vii. Values of a row all relate to some thing or portion of a thing.
viii. Repeating groups (collections of logically related attributes that occur multiple times within one record occurrence) are not allowed.
ix. Duplicate rows are not allowed (this can be prevented through candidate keys).
x. Only single valued cells are allowed. Single valued means the following:

- Cannot contain multiple values such as 'A1,B2,C3'.
- Cannot contain combined values such as 'ABC-XYZ' where 'ABC' means one thing and 'XYZ' another

xi. The logical design of the database is called Database schema.

- The Schema of a Relation: R (A1, A2, ...An)
- Relation schema R is defined over attributes A1, A2,… ...,An

The data in the database at a given instant in time is called Database instance

A relation may be expressed using the notation R(A,B,C, ...) where:

- R implies the name of the relation.
- (A,B,C, ...) imply the attributes within the relation.
- A = the attribute(s) which form the primary key.

1.1.3.2 Keys

i. *A simple* key contains only a single attribute.

ii. A *composite key* is a key that contains more than one attribute.

iii. A *candidate key* is an attribute (or set of attributes) that uniquely identifies a row. A candidate key must possess the following properties:

- Unique identification - The value of the key must uniquely identify that row for every row.

- Non redundancy - No attribute in the key can be discarded without destroying the property of unique identification.

iv. The candidate key which is selected as the principal unique identifier is called the *primary key*. Every relation must contain a primary key. It is usually the key selected to identify a row when the database is physically implemented. For example, a part number is selected instead of a part description.

v. Any set of attributes that uniquely identifies a row is called a super key. A superkey differs from a candidate key in that it does not require the non redundancy property.

vi. A foreign key is an attribute (or set of attributes) that appears (usually) as a non key attribute in one relation and as a primary key attribute in another relation. It is possible for a foreign key to be the whole or part of a primary key:

- A many-to-many relationship can only be implemented by introducing an intersection or link table which then becomes the child in two one-to-many relationships. The intersection table therefore has a foreign

key for each of its parents, and its primary key is a composite of both foreign keys.

- A one-to-one relationship requires that the child table has no more than one occurrence for each parent, which can only be enforced by letting the foreign key also serve as the primary key.

vii. A semantic or natural key is a key for which the possible values have an obvious meaning to the user or the data. For example, a semantic primary key for a COUNTRY entity might contain the value 'USA' for the occurrence describing the United States of America. The value 'USA' has meaning to the user.

viii. A key for which the possible values have no obvious meaning to the user or the data is known as a technical or *surrogate* or *artificial key* . These are used instead of semantic keys for any of the following reasons:

- When the value in a semantic key is likely to be changed by the user, or can have duplicates. For example, on a PERSON table it is unwise to use PERSON_NAME as the key as it is possible to have more than one person with the same name, or the name may change such as through marriage.
- When none of the existing attributes can be used to guarantee uniqueness. In this case adding an attribute whose value is generated by the system, e.g from a sequence of numbers, is the only way to provide a unique value. Typical examples would be ORDER_ID and INVOICE_ID. The value '12345' has no meaning to the user as it conveys nothing about the entity to which it relates.

A key functionally determines the other attributes in the row, thus it is always a determinant.

1.1.3.3 Relationships

Relationships in the real world are modelled through foreign keys. Relationships between real-world entities can be quite complex, involving numerous entities each having multiple relationships with each other. In a relational database such as

Microsoft Access, only relationships between pairs of tables are considered. These tables can be related in one of three different ways: one-to-one, one-to-many or many-to-many.

- **One-to-One Relationships**

Two tables are related in a one-to-one (1—1) relationship if for every row in the first table, there is at most one row in the second table. For example, one might keep most patient information in tblPatient, but put especially sensitive information (e.g., patientname, social security number and address) in tblConfidential.

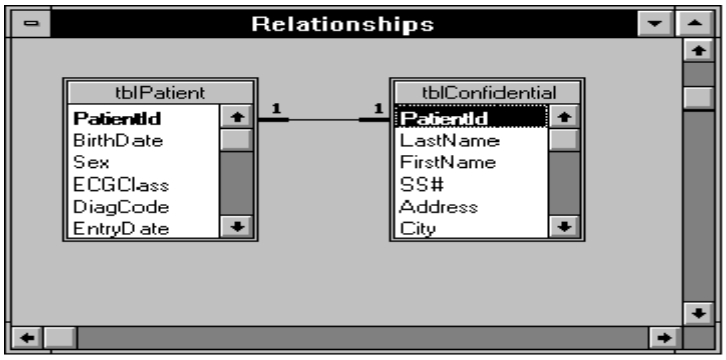


Figure 1.4 One-to-One Relationships [2]

Tables that are related in a one-to-one relationship should always have the same primary key, which will serve as the join column.

- **One-to-Many Relationships**

Two tables are related in a one-to-many (1—M) relationship if for every row in t the first table, there can be zero, one, or many rows in the second table, but for every row in the second table there is exactly one row in the first table. For example, each order for a pizza delivery business can have multiple items. Therefore, tblOrder is related to tblOrderDetails in a one-to-many relationship.

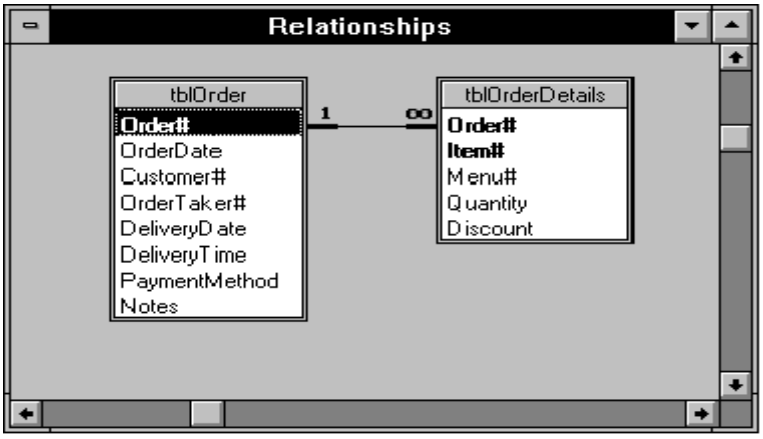


Figure 1.5: One to Many Relationships [2]

- **Many-to-Many Relationships**

  Two tables are related in a many-to-many (M—M) relationship when for every row in the first table, there can be many rows in the second table, and for every row in the second table, there can be many rows in the first table. Many-to-many relationships can't be directly modeled in relational database programs, including Microsoft Access. These types of relationships must be broken into multiple one-to-many relationships.

1.1.3.4 Drawbacks of the Relational Database

Few drawbacks of relational databases are:

- Generally, the relational model works best when there are a relatively small and static number of relationships between objects.
- It is not helpful when the data model evolves over time.
- Relational Databases are not suitable for semantic search

All these limitations of relational databases led to the invention of graph databases. Graph databases are usually schema-less and allow a set of nodes (object instances) with dynamic properties (corresponding to columns or attributes) to be arbitrary linked to other nodes through edges (associations).

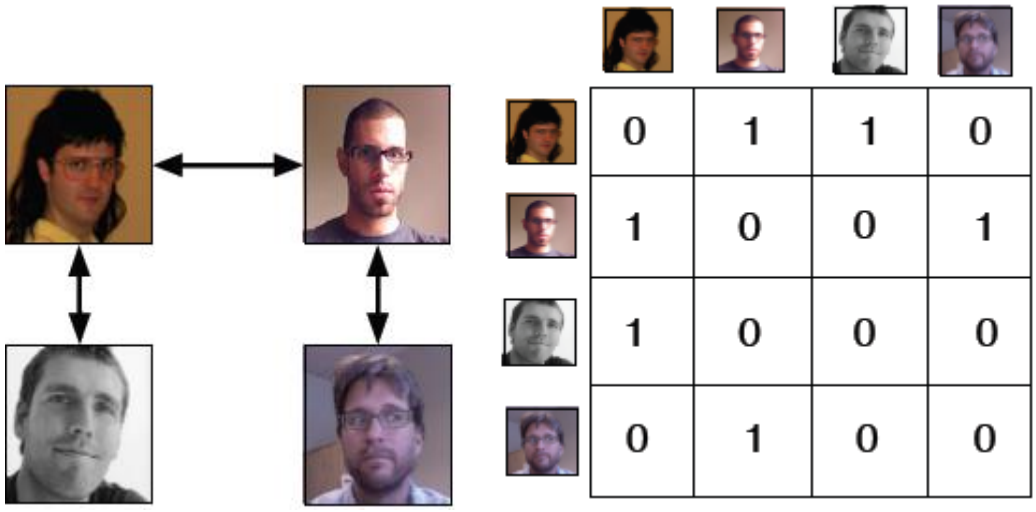
1.2 Graph

A graph is a data structure that links a set of vertices by a set of edges. These elements make direct reference to one another, and as such, there is no notion of a join operation. The direct references between graph elements make the joining of data explicit within the structure of the graph .A graph is represented by G = (V;E) where V is the set of vertices and E is the set of edges.

1.2.1 The Single-Relational, Directed Graph as a Matrix

A single-relational graph defined as G = (V;E □ (V × V)) can be represented as the adjacency matrix where

A(i;j) = (1 if (i; j) □ E 0otherwise:)



. **G** **A**

Figure 1.6 Single Relational Graph and Matrix [3]:

- All vertices are homogenous, *i.e.* vertices denote the same type of object (e.g. people, webpages, etc.).
- All edges are homogenous, *i.e.* edges denote the same type of relationships (e.g. friendship, works with, etc.).

1.2.2 Applications of Single-Relational Graphs

- Social: define how people interact (collaborators, friends, kins).
- Biological: define how biological components interact (protein, food chains, gene regulation).
- Technology: define the connectivity of Internet routers, web pages, etc.

Language: define the relationships between words.

1.2.3 The Limitations of Single-Relational Graph Modeling

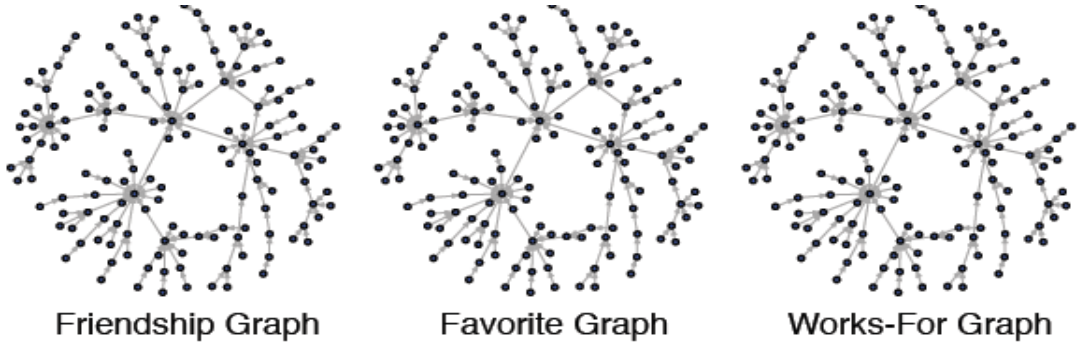


Figure 1.7 Single Relational Graph Exmaples [4]

Unfortunately, single-relational graphs are independent of each other. This is because G = (V;E) there is only a single edge set E (i.e. a single type of relation).

1.2.4 Multi-Relational Graph

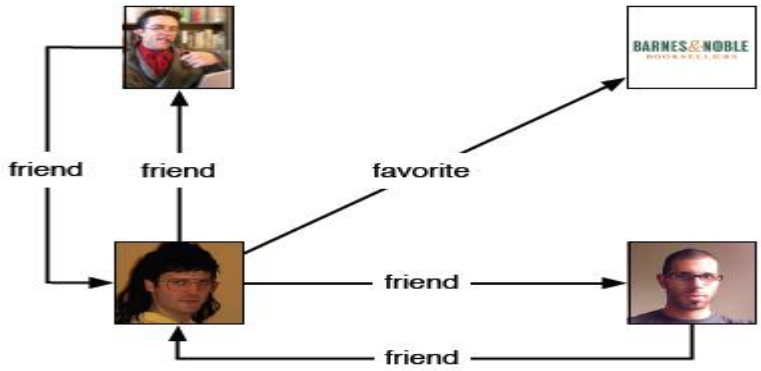


Figure 1.8 Multi Relational Graph [3]

An edge is added (i; l) € $E_{favorite}$. Now there are multiple types of relationships: E_{friend} and $E_{favorite}$ (2 edge sets).

1.2.5 Property Graph

A property graph denotes vertices (nodes, dots) and edges (arcs, lines). Edges in a property graph are directed and labeled/typed (e.g. "marko *knows* peter"). Both vertices and edges (known generally as elements) can have any number of key/value pairs associated with them. These key/value pairs are called properties.

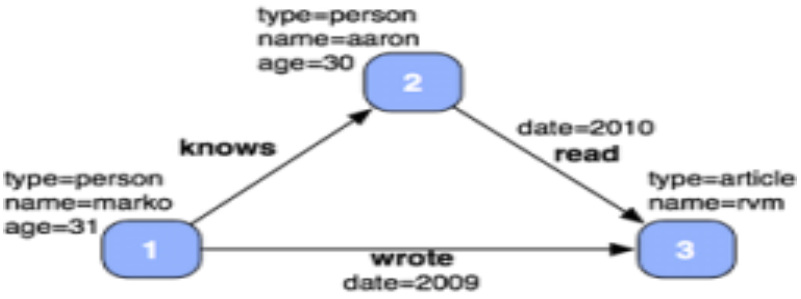


Figure 1.9 Property Graph [4]

1.3 Graph Databases

Unlike other databases which store their data in rows and columns, a graph database stores all information in a network of nodes and edges. Edges represent the connection between nodes which effectively represent objects. Because Edges and Nodes are represented as objects (like objects developers are used to) one can assign attributes (sometimes called properties) to them. Adding a direction to an edge ultimately creates a so-called property graph which represents the explicit structure of data inside a graph database.

So, unlike other database approaches, a graph database explicitly represents a graph. While other databases need to use indices and relational helpers (like relational tables which are coupled using JOINs) a graph database can traverse from one object to the next objects.

There are several implementations of graph database. Both nodes and edges can have properties that depict their specific characteristics. Some of the best known graph databases are: Infogrid, HypergraphDB, Jena, DEx, FlockdB and Neo4j[5, 6, 7].

With the exponential rapid development of Internet and its advent as an indispensable medium for electronic sharing of data, it has emerged as a huge repository of unstructured data. This has resulted in making the data search and knowledge extraction, a very cumbersome task if one continues using the legacy relational databases. The retrieval of relevant information from internet in minimum possible time and with maximum efficiency has been possible with the help of graph databases. Easily retrieving the relevant search results from the vast ocean of information requires the search to be semantic and not syntactic.

1.4 Structure of the Thesis:

The rest of the thesis is organized in the following order:

Chapter 2 - Provides a review of all the work done in the area of graph databases

Chapter 3 - Gives the problem statement and methodology used to solve it.

Chapter 4 – Explains the graph databases in detail.

Chapter 5 - Explains the experiments performed and evaluates the results achieved.

Chapter 6 - Presents the conclusion of this thesis and suggestions for future work.

Thesis concludes with references.

Chapter 2

Literature Review

Information Storage has been a challenge throughout human history and existed long before modern computer systems. With examples like Dewey Decimal Classification (also called the Dewey Decimal System) , a proprietary system of library classification developed by Melvil Dewey in 1876 made information retrieval and indexing more efficient. However it still required vast amounts of physical volume to store data and relied on the human intellect to process trivial relations in that data.

In 1964 , Charles Bachmann developed the first commercial database management system (DBMS) , IDS - Integrated Data Store based upon an early network data model while working at Honeywell [8]. IDS maintained a single set of shared files on disk. Programs responsible for particular tasks, such as billing or inventory updates, retrieved and updated these files by sending requests to IDS. IDS provided application programmers with a set of powerful commands to manipulate data.

In the late 1960s, IBM and North American Aviation (later Rockwell International) developed the first commercial hierarchal DBMS, IMS - Information Management System, and its DL/1-language. These models lack a good abstraction level which implies that it is difficult to separate the db-model from the actual implementation. The data structures provided are not apt for modelling non traditional applications. They permit database navigation at the record level by providing low-level operations that can be used to derive more abstract structures. The solution to these problems was given by Codd. In 1970, Codd released "A Relational Model of Data for Large Shared Data Banks." which provided the first definition of Relational Model.

Codd's relational model suggested that all data in a database could be represented as a tabular structure (tables with columns and rows, which he called relations) and that these relations could be accessed using a high-level non-procedural (or declarative) language [reference]. Edgar F. Codd published a series of papers about relational databases from 1970 to 1972 [9,10,11] in which he focused on the concept of abstraction levels by introducing a separation between the physical and logical levels. It was a major development because it gave the data modelling discipline a mathematical foundation. It is based on the simple notion of relation, which together

with its logic and algebra, made it the most widely used model and its standard query language SQL, became a paradigmatic language for querying. But with time, database community realized that the relational model was inadequate for data intensive domains (knowledge bases, engineering applications) involving complex data objects and object interactions such as CAD/CAM software, computer graphics, and information retrieval.

In order to provide additional semantics necessary to model these new applications Object-oriented (O-O) db-models appeared in the eighties [12]. According to Rao in 1994, "The object-oriented database (OODB) paradigm is the combination of object-oriented programming language (OOPL) systems and persistent systems". O-O db-models view the world as a set of complex objects having certain states (data), where interaction is via method passing. Although O-O db-models permit much richer structures than the relational db-model, they still require that all data conform to a predefined schema. One of the promising solutions to this problem was given by document databases such as Lucene, which is able to fully index large document collections and support queries that rank the documents according to information retrieval measures.

But what makes document databases really different, is the fact that documents are usually retrieved through dynamic and unpredictable queries. Thus document databases can usually associate any number of fields of any length to a document. Document databases are usually schema-less, i.e. there is no predefined data model. But with the vast increase in the usage of internet, these document models were not proving to be efficient in performing many operations for example the World Wide Web exhibits far more complicated networks of relationships than were expected . The network of hyperlinks connecting all the pages on the World Wide Web is highly complex and almost impossible to model efficiently in a document database. Similar issues are involved in modeling the social network like Twitter, Facebook, etc. Implementing such a large network of relationships in the form of these conventional databases was an impossible task. So there was a need of the kind of model where both data manipulation and representation are graph-based.

The database community became aware of this need and proposed graph databases. A survey of the literature on community structure [13,14,15], link prediction [16,17,18] and social influence [19,20,21] shows that graphs are the predominant model for social networks. Adrian Silvescu in his paper on graph databases showed how graph

databases easily handle changes in the schema. A survey on the literature of graph databases shows that although activities around graph databases flourished in the first half of the nineties, the topic almost disappeared in later years. The reason behind it was the emergence of XML that captured all the attention of those working on hypertext which forced people working on graph databases to move to particular applications like spatial data, Web, and documents; the tree-like structure was enough for most applications at that time.

A number of papers have been published on graph databases during the 70's and 80's in early approach, Roussopoulos and Mylopoulos proposed a semantic network to store data about the database in 1975 [22]. Shipman in 1981 presented an implicit structure of graphs for the data itself in the Functional Data Model, whose goal was to provide a "conceptually natural" database interface. Logical Data Model (LDM) developed by Kuper and Vardi in 1984 [23], was proposed on a different approach according to which an explicit graph db-model intended to generalize the relational, hierarchical and network models. Later Kunii in 1987 [24] proposed a graph db-model for representing complex structures of knowledge called G-Base.

In the late eighties, Lecluse in 1988 introduced O2 [25], an object-oriented db-model based on a graph structure. Along the same lines, GOOD developed by Gyssens in 1990 was a system in which manipulation as well as representation are transparently graph based. Among the subsequent developments based on GOOD are: GMOD by Andries in 1992, which proposes a number of concepts for graph-oriented database user interfaces; Gram by Amann and Scholl in 1992, which is an explicit graph db-model for hypertext data [26]; PaMaL by Gemis and Paredaens in 1993, which extends GOOD with explicit representation of tuples and sets; GOAL developed by Hidders and Paredaens in 1993, which introduces the notion of association nodes; G-Log in 1995, proposed a declarative query language for graphs; and GDM developed by Hidders in 2005 which incorporates representation of n-ary symmetric relationships [27].

There were proposals that used generalization of graphs with data modeling purposes. Levene and Poulovassilis in 1990 introduced a db-model based on nested graphs, called the Hypernode Model, on which subsequent work was developed by Poulovassilis and Levene in 1994 and Levene and Loizou in 1995. The same idea was used for modelling multi-scaled networks and genome data [Graves et al. 1995a]. GROOVY is an object-oriented db-model which is formalized using

hypergraphs. There are several other proposals that deal with graph data models. Guting in 1994 proposed GraphDB, intended for modeling and querying graphs in object-oriented databases and motivated by managing information in transport networks. Database Graph Views [Guti´errez et al. 1994] proposed an abstraction mechanism to define and manipulate graphs stored in either relational object-oriented or file systems. The project GRAS [Kiesel et al. 1996] uses attributed graphs for modeling complex information from software engineering projects. The well known OEM [Papakonstantinou et al. 1995] model aims at providing integrated access to heterogeneous information sources, focusing on information exchange.

One more problem that arose with the evolution of Internet was the need of searching and accessing information from the WWW. A search engine [28] is a document retrieval system designed to help find information stored in a computer system, such as on the World Wide Web, inside a corporate or proprietary network, or in a personal computer. But the presence of huge amount of resources on the Web poses a serious problem of accurate search. This is mainly because today‟ s Web is a human-readable Web where information cannot be easily processed by machine. Highly sophisticated, efficient keyword based search engines that have evolved today have not been able to bridge this gap [28]. This huge repository of unstructured data has resulted in making the data search and knowledge extraction, a very cumbersome task if one continues using the legacy relational databases. One of the proposed solutions is to shift to the graph databases as they not only makes it easier to store and retrieve data for the world wide web but also provide relevant and streamlined search results from the abundant information present on the internet.

Although graph databases have existed as a data model for at least 30 years, but only recently with the revival of artificial intelligence and the emergence of the semantic web have they become more popular and on the verge of becoming a serious industry player. Among them are metadata representation models like RDF and ontology representation models like OWL, an influential graph-oriented object model. Then came the hyper graph which was designed for artificial intelligence and semantic web projects. HypergraphDB stores all the graph information in the form of key-value pairs. Another graph that came around the same time is Neo4j which was released in February 2010 and is a high performance, robust and scalable graph database solving queries with multiple relationships storing data in the nodes and relationships[29]. Neo4j was fully written in Java and can be deployed on multiple systems [30].

Neo4j's founder Emil Eifrem claims that socially enabled applications are gravitating towards graph databases because other types of databases are not effective for managing relationships between millions of users with multiple connections. Neo4j is the ideal solution for any application that relies on the relationships between records.

Many graph databases are large and growing rapidly in size. For example, the social networking site Facebook contains a large network of registered users and their friendships. The number of Facebook users has grown from less than 5 million in September 2005 to close to 10 million in September 2006, then to 50 million in September 2007. There is a critical need for efficient and effective graph querying systems to query and mine these growing graph databases.

Although the effective usage of Graph databases has been growing manifold, the problem faced is in querying the graph databases efficiently and effectively. After going through various proposals presented for graph databases, it has been realized that out of all the databases available, Neo4j is apt for representing and retrieving data for the World Wide Web.

This thesis evaluates the performance of Graph databases in comparison to relational databases by designing graph databases in Neo4j. Here, a comparison of query retrieval times of relational and Neo4j database has been conducted. The evaluation measures not only include the query retrieval times but scalability, flexibility and maturity and accuracy in search results as well in minimum time.

Chapter 3

Problem Statement

3.1 Problem Statement

During the literature review, it was analyzed that in today's environment, the relational representations were not efficient in performing many operations. With the exponential development of Internet and its advent as an indispensable medium for electronic sharing of data, it has emerged as a huge repository of unstructured data. The World Wide Web exhibits far more complicated networks of relationships than were expected when SQL was designed. The network of hyperlinks connecting all the pages on the World Wide Web is highly complex and it is almost impossible to model efficiently in a relational database.

Similar issues are involved in modelling the social network like Twitter, Facebook, etc. Implementing such problems in relational databases involves large number of joins which was expensive to be calculated. This has resulted in making the data search and knowledge extraction a very cumbersome task if one continues using the legacy relational databases. There is a need of the data store that can give streamlined and relevant results. The retrieval of relevant information from internet in minimum possible time and with maximum efficiency has thereby turned up as a major challenge.

So, graph databases can serve as an alternate for relational databases for determining relationships optimally and quickly in the rapid growth growing internet and social network. These databases can be of great help to the companies struggling with traditional databases and want some new database that can replace these legacy databases and can be used for commercial purposes like website link structures and social networking.

i. A free download software Neo4j is used to store the closely connected data items as seen in social networks.

ii. A thorough comparison of traditional relational databases is done with graph databases using parameters like query retrieval times, scalability, flexibility, maturity and accuracy in search results.

3.2 The Methodology used to solve the problem

The step-by-step methodology to be followed in storing, retrieving and analyzing of a social network is given below:

- Installing Neo4j and MySQL and enter the data using link structure like social network in both the databases.
- Run similar queries on both databases first for hundred and then for five hundred objects with the help of cypher query in Neo4j and PHP in MySQL.
- Analyze the relevance of results, query execution time and scalability as given by both the databases.

Chapter 4
Graph Databases

4.1 Introduction to Graph Databases

Relational database models are providing the storage support for many applications. Relational database systems store biological datasets, economic transactions, provide storage for dynamic websites, etc. Although the relational model has proven efficient and scales well for large datasets in table form, it is not adequate for applications that deeply analyze relationships among entities [31]

Similar issues are involved in modelling the social network like Twitter, Facebook, etc. Implementing such problems in relational databases involves large number of joins which was expensive to be calculated. This has resulted in making the data search and knowledge extraction a very cumbersome task if one continues using the legacy relational databases. There is a need of the data store that can give streamlined and relevant results. The retrieval of relevant information from internet in minimum possible time and with maximum efficiency has thereby turned up as a major challenge. So, graph databases can serve as an alternate for relational databases for determining relationships optimally and quickly in the rapid growth growing internet and social network.

Graph database models can be defined as those in which data structures for the schema and instances are modelled as graphs or generalizations of them, and data manipulation is expressed by graph-oriented operations and type constructors. These models took off in the eighties and early nineties alongside object oriented models. Recently, the need to manage information with graph-like nature has re-established the relevance of this area [32].

4.2 Advantages Of Graph Databases

Graph databases can be of great help to the companies struggling with traditional databases and want some new database that can replace these legacy databases. The

main issues of relational databases that could be easily resolved by graph databases are:

- **Generally, the graph model works best even when there are millions of relationships between objects.**

  Relational Databases have been providing the storage support for many decades now with implementations like Oracle, MySQL, etc. Way back people used database just for storing tabular data like purchase reports and finance records. Relational databases were perfect as one could associate a transaction in finance table with an item in purchase table. But in today's environment, these relational representations are not efficient in performing many operations for example, With the intense increase in usage of internet leading to need for storing large amounts of interconnected data, there was a clear desire for a data store tailored to the needs of graph data. Graph databases are optimized for these types of networks (social networking and website link structure), as graph is a natural way of storing connections between users.

- **It can easily accommodate the schema changes**

  The reason the relational database doesn't represent knowledge very well is that the relational database is only good at storing objects and relationships between them when one fully understands exactly what objects and what relationships will be managed upfront. When one needs to represent some new type of relationship between the objects in a relational database, it tends to fail, or be very difficult. In fact, the relational database is not even particularly good at adding new types of objects to the database. Most relational databases actually have an upper limit on the types of objects, typically referred to as tables, which can be handled. Too many tables in a database schema is considered as a bad design.

- **Graph Databases are suitable for semantic search**

The Internet revolutionised the way in which our society disseminates and uses information. The popularity of the Internet resulted in an exponential growth of information available on the Internet and the associated Web. A consequence of this

popularity is an overload of information available on almost any topic which in turn has created several new problems for users. How to find relevant information in such a short time became the key constraint to the process of human society progress while facing the vast ocean of information. There was a need of the data store that could give streamlined and relevant results.

4.3 Applications Of Graph Database

Several areas have witnessed the emergence of huge data networks called complex networks. So graph databases are the best database to implement such complex network of relationships having million sof nodes and relationships. The main application areas of graph databases are:

i. Social networks

In social networks, nodes are people or groups, while links show relationships or flows among nodes. Some examples are friendships, business relationships, research networks (collaboration, coauthorship), communication records (mail, telephone calls, email), computer networks, and national security There is growing activity in the area of social network analysis and also in visualization and data processing techniques for these networks.

ii. Information networks

Information networks model relations representing information flow, such as citations among academic papers, World Wide Web (hypertext, hypermedia), peer-to-peer networks, relations among word classes in a thesaurus, and preference networks.

iii. Technological networks

In technological networks, the spatial and geographical aspects of the structure are dominant. Some examples are the Internet (as a computer network), electric power grids, airline routes, telephone networks, delivery network (post office), and Geographic Information Systems (GIS) are today covering a big part of this area (roads, railways, pedestrian traffic, rivers).

iv. Biological networks

Biological networks represent biological information whose volume, management and analysis has become an issue due to the automation of the process of data gathering. A good example is the area of genomics, where networks occur in gene regulation, metabolic pathways, chemical structure, map order and homology relationships among species. There are other kinds of biological networks, such as food webs and neural networks.

It is important to stress that classical query languages offer little help when dealing with the type of queries needed in these areas. As examples, data processing in GIS include geometric operations (area or boundary, intersection, inclusions, etc), topological operations (connectedness, paths, neighbours, etc) and metric operations (distance between entities, diameter of the network, etc). In genetic regulatory networks, examples of measures are: connected components (interactions between proteins), and nearest neighbour degrees (strong pair correlations). In social networks, some measures are: distance, neighbourhoods, clustering coefficient of a vertex, clustering coefficient of a network, between ness, and size of giant connected components, and size distribution of finite connected components Similar problems occur in the Semantic Web, where RDF queries would benefit from being able to reason about graph aspects.

4.4 Graph Database Libraries

I. HypergraphDB

Hypergraph is a GDB, which was designed for artificial intelligence and semantic web projects [33]. As a consequence of the requirements from these environments, HypergraphDB stores not only graphs but also hypergraph structures. A hypergraph is a mathematical generalization of the concept of a graph, in which the edges are substituted by hyperedges. The difference between edges and hyperedges is the number of nodes that they connect: a regular edge connects two nodes of a graph, but a hyperedge connects an arbitrary set of nodes.

II. **DEX**

DEX is an efficient GDB implementation based on bitmap representations of the entities. All the nodes and edges are encoded as collections of objects, each of which has a unique oid that is a logical identifier [34]. DEX implements two main types of structures: bitmaps and maps. DEX converts a logical adjacency matrix into multiple small indexes to improve the management of out-of-core workloads, with the use of efficient I/O and cache policies. DEX encodes the adjacency list of each node in a bitmap, which for the adjacent nodes has the corresponding bit set. Given that bitmaps of graphs are typically sparse, the bitmaps are compressed, and hence are more compact than traditional adjacency matrices.

III. **Jena**

The Resource Description Framework (RDF) is a standard for describing relationships among entities. A relationship expressed in RDF is expressed as a triplet: a subject, a predicate and an object. The interpretation is that the subject applies a predicate on the object. Therefore, a RDF description can be viewed as a graph where the subject and the object are the vertices, and the predicate corresponds to the edge that relates them. There are several RDF based graph implementations: Jena [35], Sesame [36], AllegroGraph etc.

4.5 Neo4j

Neo4j is a high-performance, NOSQL graph database with all the features of a mature and robust database. The programmer works with an object-oriented, flexible network structure rather than with strict and static tables — yet enjoys all the benefits of a fully transactional, enterprise-strength database.

Neo4j is an open source project available in a GPLv3 Community edition, with Advanced and Enterprise editions available under both the AGPLv3 and commercial licenses, supported by Neo Technology.

Neo4j is fully written in Java and can be deployed on multiple systems [4]. It is comprised of two parts

-A client that sends commands to the server via RMI.

-A Server that processes these commands and sends back the processed result to the client.

Neo4j graph consist of:

- Nodes that are connected by
- Relationships, with
- Properties on both nodes and relationships.

i. Nodes

The fundamental units that form a graph are nodes and relationships. In Neo4j, both nodes and relationships can contain properties. Nodes are often used to represent *entities*, but depending on the domain relationships may be used for that purpose as well.

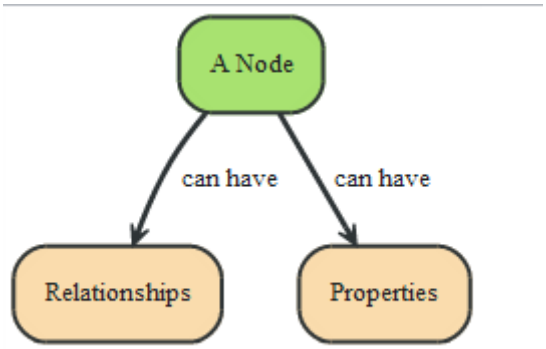


Figure 4.1 Nodes in Neo4j [38]

ii. Relationship

Relationships between nodes are a key part of a graph database. They allow for finding related data. Just like nodes, relationships can have properties.

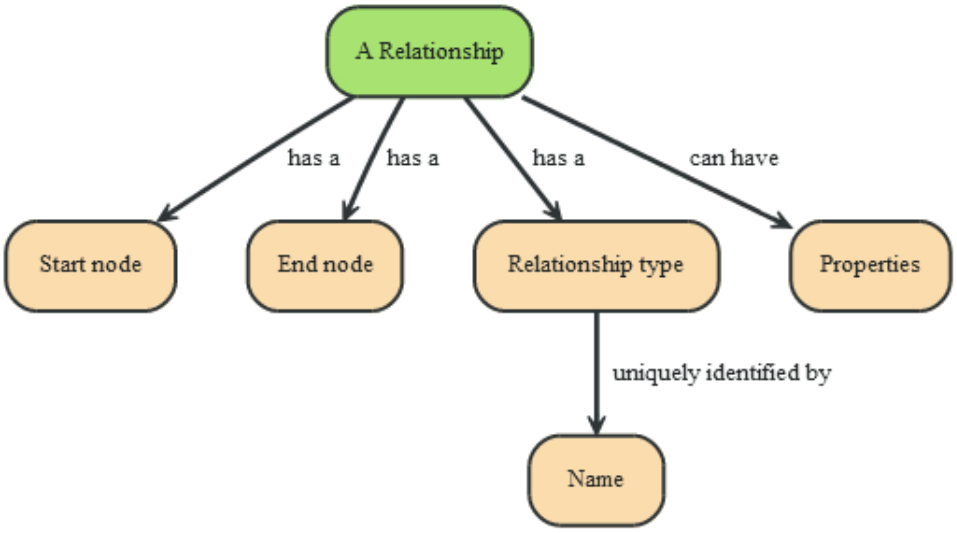


Figure 4.2 Relationships in Neo4j [39]

A relationship connects two nodes, and is guaranteed to have valid start an end nodes.

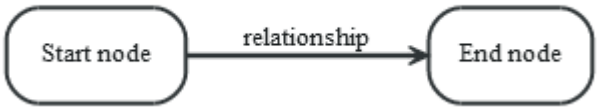


Figure 4.3 Relationships and nodes in Neo4j [39]

As relationships are always directed, they can be viewed as outgoing or incoming relative to a node, which is useful when traversing the graph:

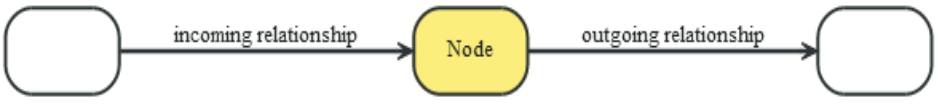


Figure 4.4 Incoming And Outgoing Relationship [39]

iii. Properties

Both nodes and relationships can have properties. Properties are key-value pairs where the key is a string. Property values can be either a primitive or an array of one primitive type.

4.5.1 Cypher Query

The database is queried with cypher Query Language. It is a new query language that has been recently added to the Neo4j. Unlike imperative languages like Java and scripting language like Gremlin and the Ruby Neo4j bindings, Cypher is a declarative language. Using Cypher, efficient querying of the graph is possible, as there is no need to write traversers in the code. Cypher is still growing and maturing, and that means that there probably will be breaking syntax changes. It also means that it has not undergone the same rigorous performance testing as the other components. Cypher is designed to be a humane query language, suitable for both developers and (importantly, we think) operations professionals who want to make ad-hoc queries on the database. Its constructs are based on English prose and neat iconography, which helps to make it (somewhat) self-explanatory. Cypher is designed to be a humane query language, suitable for both developers and perations professionals who want to make ad-hoc queries on the database. Its constructs are based on English prose and neat iconography, which helps to make it self-explanatory.

Cypher is inspired by a number of different approaches and builds upon established practices for expressive querying. Most of the keywords like `WHERE` and `ORDER BY` are inspired by SQL. Pattern matching borrows expression approaches from SPARQL. Regular expression matching is implemented using the Scala programming language.

The query language is comprised of several distinct parts.

- START: Starting points in the graph, obtained by element IDs or via index lookups.
- MATCH: The graph pattern to match, bound to the starting points in START.
- WHERE: Filtering criteria.
- RETURN: What to return.

The detailed description of these keywords in cipher query is as given:

i. Start

Every query describes a pattern, and in that pattern one can have multiple start points. A start point is a relationship or a node that form the starting points for a pattern match. One can either introduce start points by id, or by index lookups.

ii. Match

Pattern matching is one of the pillars of Cypher. The pattern is used to describe the shape of the data that one is looking for. Cypher will then try to find patterns in the graph — these are called matching sub graphs.

The description of the pattern is made up of one or more paths, separated by commas. A path is a sequence of nodes and relationships that always start and end in nodes. An example path would be: `(a)-->(b)`

Paths can be of arbitrary length, and the same node may appear in multiple places in the path. Node identifiers can be used with or without surrounding parenthesis. Patterns have bound points, or start points. They are the parts of the pattern that are already "bound" to a set of graph nodes or relationships. All parts of the pattern must be directly or indirectly connected to a start point — a pattern where parts of the pattern are not reachable from any start point will be rejected. The optional relationship is a way to describe parts of the pattern that can evaluate to null if it can not be matched to the graph. It's the equivalent of SQL outer join — if Cypher finds one or more matches, they will be returned. If no matches are found, Cypher will return a null.

It is explained in detail through the Figure 4.5. From the given graph if one wants to retrieve all nodes that are blocked by Anders.

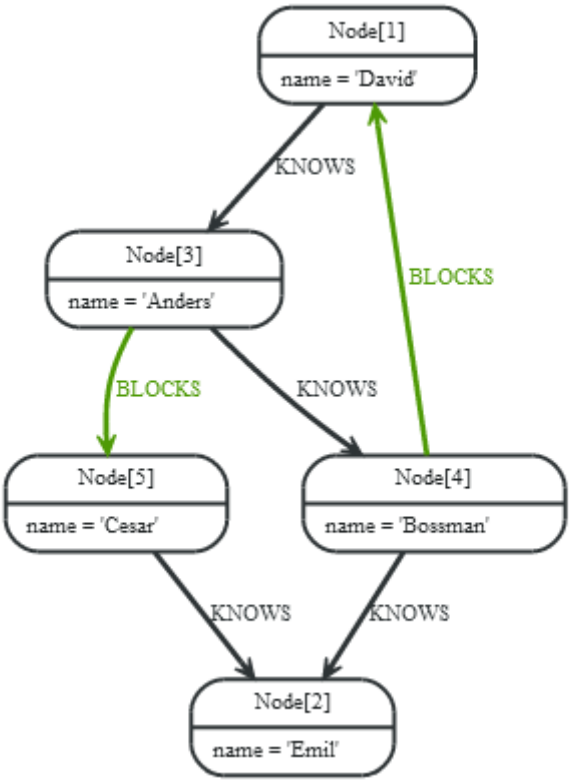


Figure 4.5 Neoj Graph Example [40]

The cypher query for the above query is:

START n=node(3) MATCH (n)-[:BLOCKS]->(x) RETURN x

The result retrieved for the above query is as given in Figure 4.6

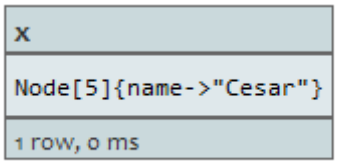


Figure 4.6Query Result1 [40],

iii. Where

If one needs filtering apart from the pattern of the data that one is looking for, clauses can be added in the where part of the query. It is explained in detail through the Figure 4.7. From the given graph if one wants to retrieve nodes where age < 30 and name = “Tobias” or name =”Tobias”

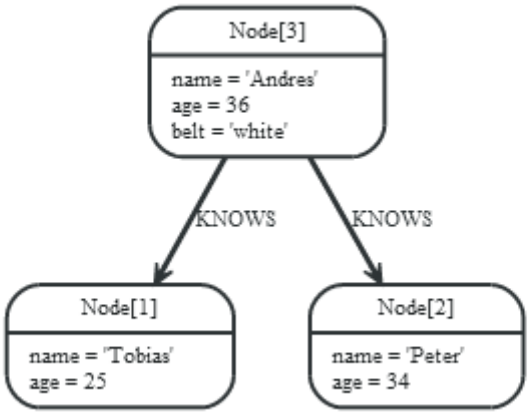


Figure 4.7] Neo4j Graph Example [40]

The cypher query for the above query is

START n=node(3) where (n.age < 30 and n.name=”Tobias”) or not (n.name = “Tobias”)

The result retrieved for the above query is as given in Figure 4.8

n
Node[3]{name->"Andres",age->36,belt->"white"}
Node[1]{name->"Tobias",age->25}
2 rows, 0 ms

Figure 4.8 Query Result 2 [40]

4.6 Semantic Web in Graph Databases

According to Tim Berners-Lee “The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [2, 27].

"The Semantic Web is a vision: the idea of having data on the Web defined and linked in such a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications" [3].

Semantic means adding meaning of data to be discovered by computers. It is a vision of a new architecture for the World Wide Web, characterized by the association of machine-accessible formal semantics with more traditional Web content. The core

idea is to create the metadata describing the data, which will enable computers to process the meaning of things. Semantic web promises to infuse the Internet with a combination of metadata, structure, and various technologies so that machines can derive meaning from information, make more intelligent choices, and complete tasks with reduced human intervention. Semantic web vision is oriented toward machine-readable resources rather than human-readable. It requires resource description so that machines know what they mean (metadata) [5]. The process of associating metadata with resources (audio, video, structured text, unstructured text, web pages, images etc) is called annotation and ***semantic annotation*** is the process of annotating resources with semantic metadata.

4.6.1 Role of Annotations in Semantic Web

Meaningful use of any data requires knowledge about its organization and content. Contextual information that establishes relationships between the data and the real world aspects it applies to is called metadata. In other words, metadata is data that describes information about a piece of data, thereby creating a context in terms of the content and functionality of that data. Broadly speaking, there are two kinds of metadata - structural and syntactic metadata [11].

Structural metadata provides information about the organization and structure of some data, e.g. format of the document. Semantic metadata on the other hand, provides information 'about' the data for example the meaning or what the data is about and the available semantic relationships from a domain model in which the data is defined. The key aspect behind the realization of the Semantic Web vision is the provision of metadata and the association of metadata with web resources.

The process of associating metadata with resources (audio, video, structured text, unstructured text, web pages, images etc) is called annotation and **semantic annotation** is the process of annotating resources with semantic metadata.

Annotation is about attaching names, attributes, comments, descriptions, etc. to a document or to a selected part in a text. It provides additional information (metadata) about an existing piece of data.

4.5.1 Implementation of Semantic Web in Graph Databases

The data model for semantic web in graph databases consists of nodes connected by labelled arcs, where the nodes represent Web resources and the arcs represent properties of these resources. It is built on the concept of a statement, a triple of the form {predicate, subject, object}. The interpretation of a triple is that <subject> has a property <predicate> whose value is <object>. a <subject> is always a resource named by a URI with an optional anchor id, <predicate> is a property of the resource, and the <object> is the value of the property for the resource [9]. Consider the following triples (Figure 4.9):

{dc:Publisher, http://www.w3.org, "World Wide Web Consortium"}

{dc:Title, http://www.w3.org, "W3C Home Page"}

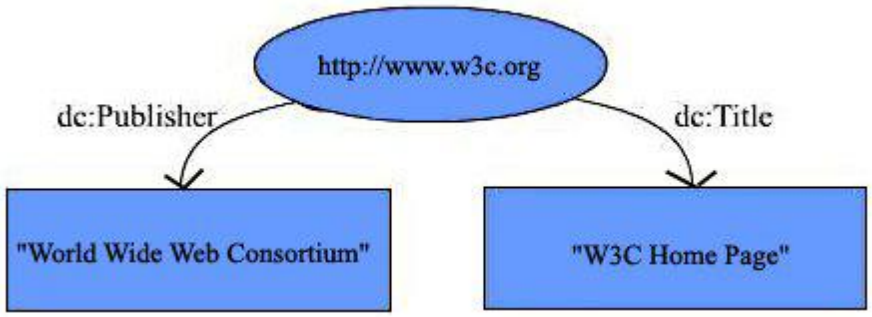


Figure 4.9 Graph Database Triple [41]

Chapter 5

Implementation and Results

5.1 Implementation of Neo4j

The evaluation between MySQL and Neo4j is based upon a set of predefined queries. Graph databases are implemented using Neo4j Community version 1.6. The database is queried with Cypher Query Language. Queries designed to analyze the performance difference between a relational database and a graph database are:

S0: Find all friends of Esha.

S1: Find the favorite movies of Esha's friends.

S2: Find the lead actors of favorite movies of Esha's friends.

Schema for graph databases is represented in Fig 5.1

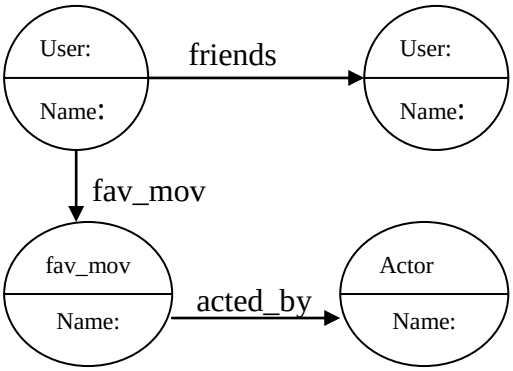


Figure 5.1 Graph Databasre Schema

Step 1:

First step in the entire implementation is to connect to a shell server. By default, the Neo4j Server is bundled with a Web server that binds to local host on port 7474, answering only requests from the local machine. To start the shell and connect to a running server, ***./bin/neo4jshell*** command is issued at the command prompt.

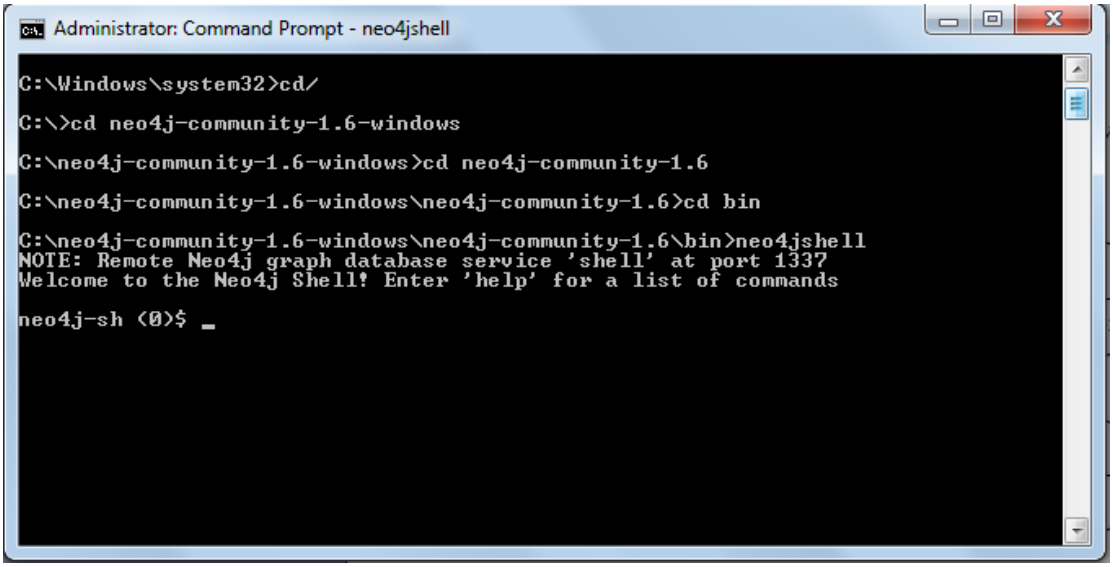


Figure 5.2 Connecting to Neo4j server

Step 2:

Once the successful connection has been made to the server, the next step is to create the graph. In Neo4j, objects are depicted as nodes. New nodes are created through this command:

mknode –cd --np –v "{'name':'charu'}" where

--cd Go to the created node

--np Properties to set for the new node

-v Verbose Mode , Display created node.

Nodes corresponding to each user, their favourite movie and the actors in the movie are created by issuing the above command for hundred nodes first and then for five hundred nodes.

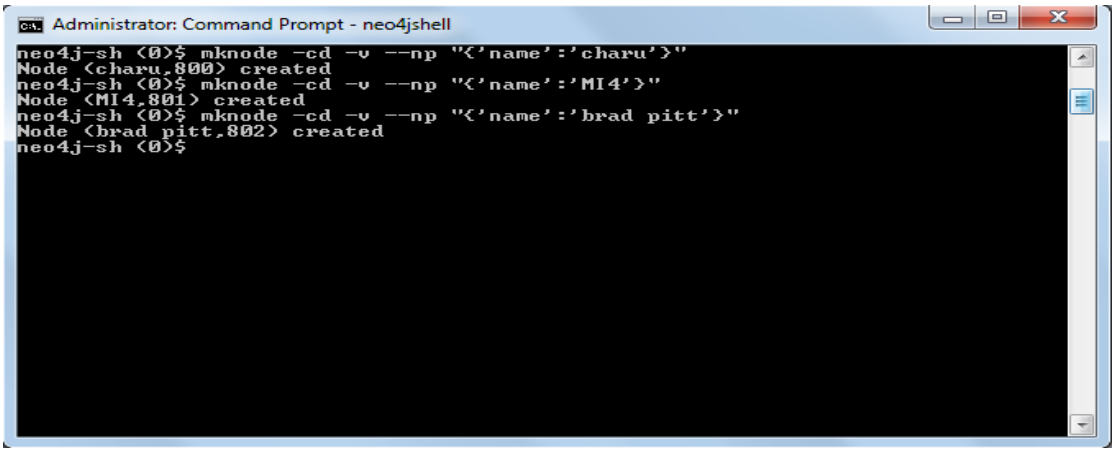


Figure 5.3 Creating nodes in Neo4j

Figure 5.4 shows few nodes created in the original graph.

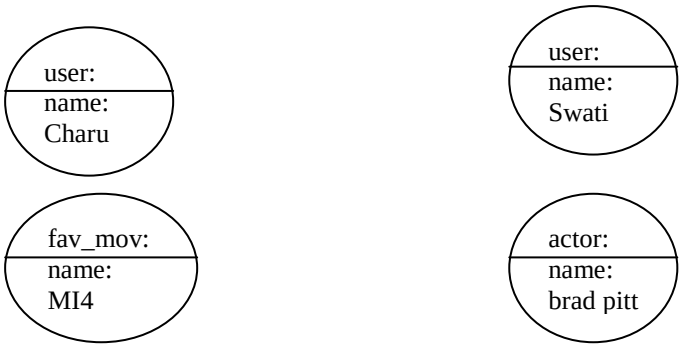


Figure 5.4 Nodes with Properties

Step 3:

Relationships between nodes is represented by edges so the next step is to add relationships between edges. Relationships are created through this command:

New relationships are added to the current or new nodes for example: (font)

mkrel –ct KNOWS <will create a relationship to a new node>

-c Supplied if there should be created a new node.

-v Verbos mode: display created nodes/relationships.

-t The relationship type.

Figure 5.5 creating relationships in Neo4j

Once the above step is completed for all the nodes and relationships, the entire graph has been generated. Sub graph of the actual graph is shown here in Figure 5.6 that shows the nodes and the kind of relationships between these nodes.

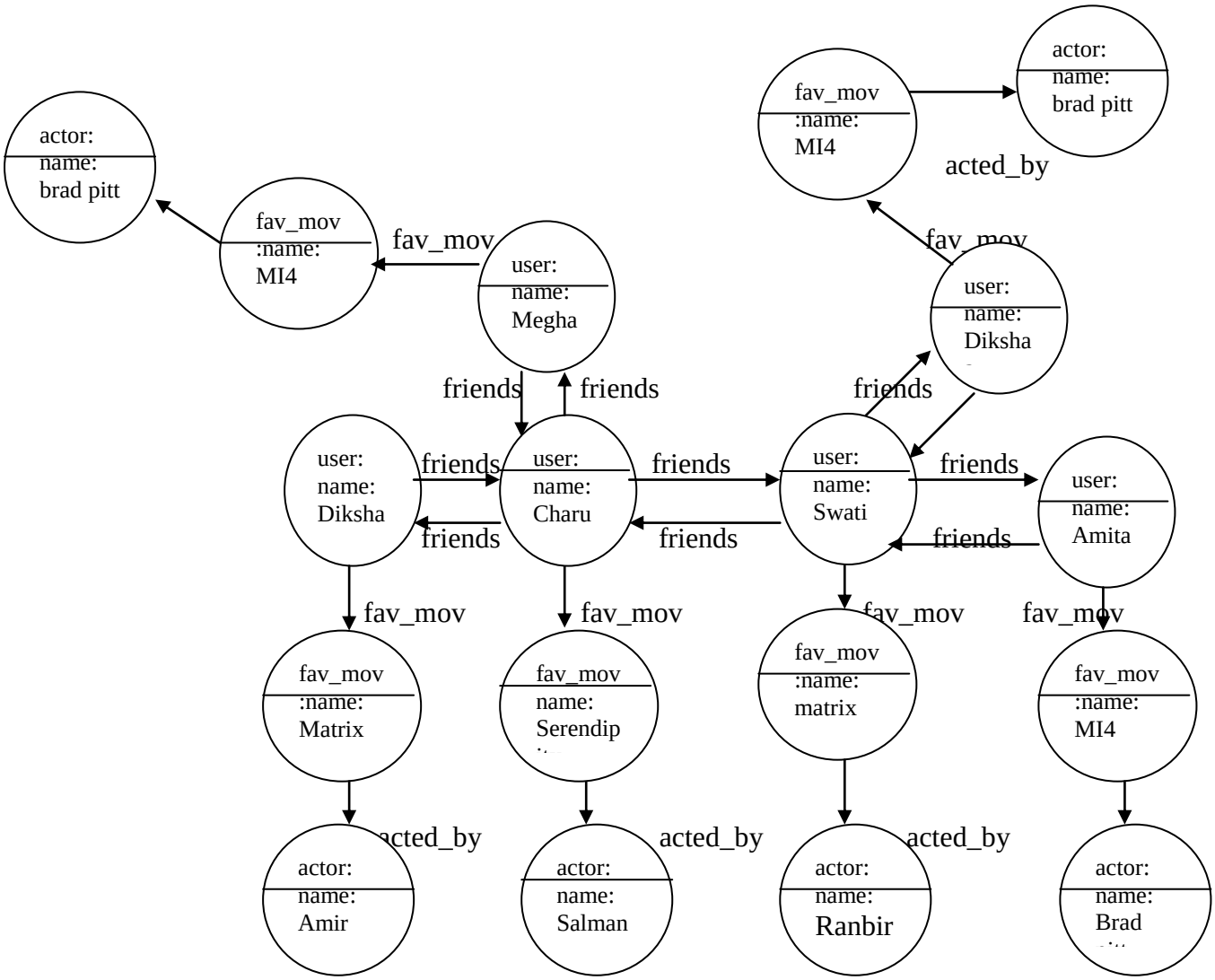


Figure 5.6 Subgraph of the actual graph

Step4

Once the graph has been made the next step is to retrieve the query results and note down the retrieval times of each query for hundred and five hundred nodes respectively. The database is queried with Cypher Query Language. Neo4j itself gives the retrieval time of the query in milliseconds. So we do not have to explicitly do the coding to find out the retrieval times of the queries.

The first query is to find all friends of Esha. The Neo4j query for the same is:

- start n=node(192) match n-[:friends]->y return y.name

Retrieval times for the above query as calculated by Neo4j for hundred and five hundred users are shown in Figure 5.7 and Figure 5.8 respectively. .

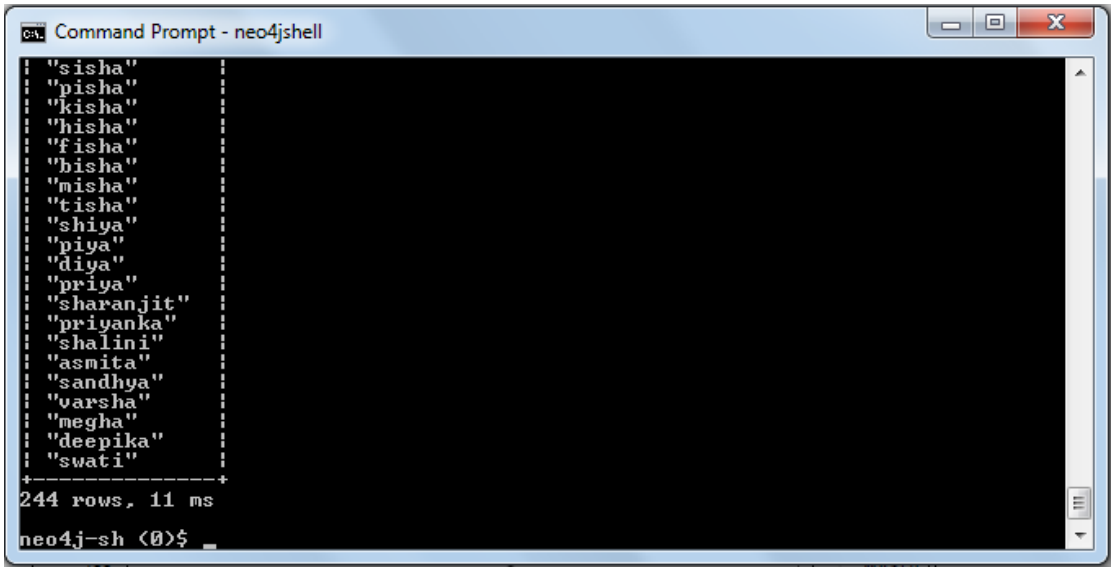


Figure 5.7 Retrieval time for 500 users for S0

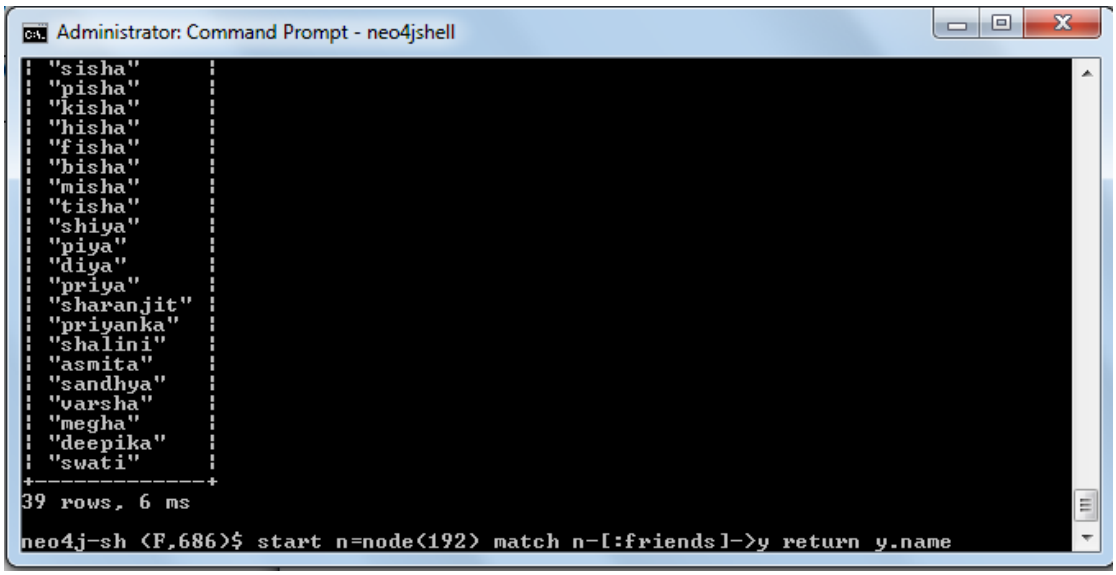


Figure 5.8 Retrieval times for 100 users FOR S0

The second query is to find the favourite movies of Esha's friends. The Neo4j query for the same is:

- start n=node(192) match n-[:friends]->y-[:fav_mov]->z return z.name

Retrieval times for the above query as calculated by Neo4j for hundred and five hundred users are shown in Figure 5.9 and Figure 5.10 respectively. .

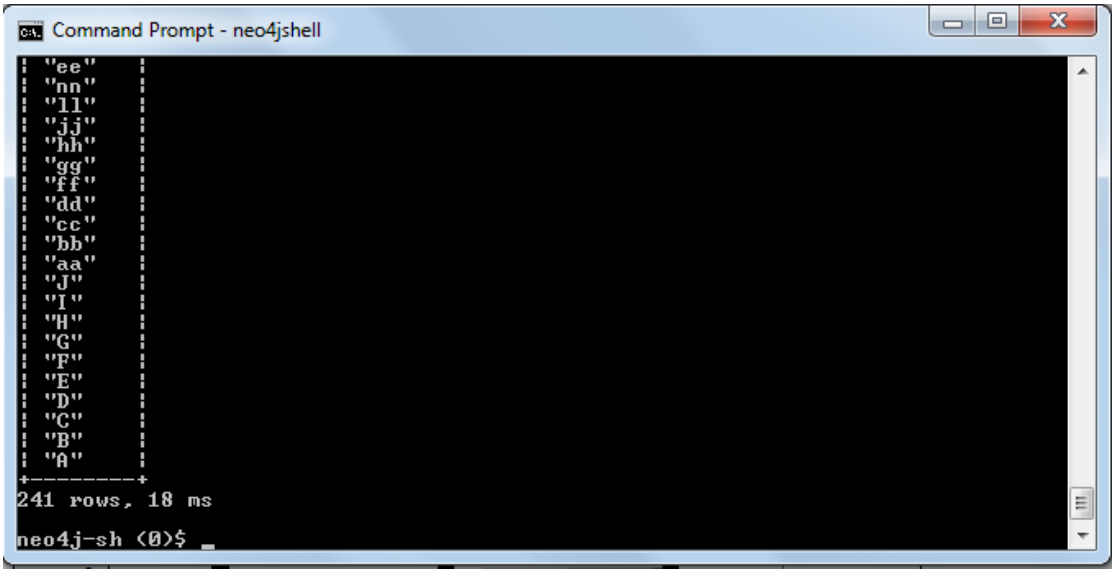


Figure 5.9 Retrieval time for 500 users for S1

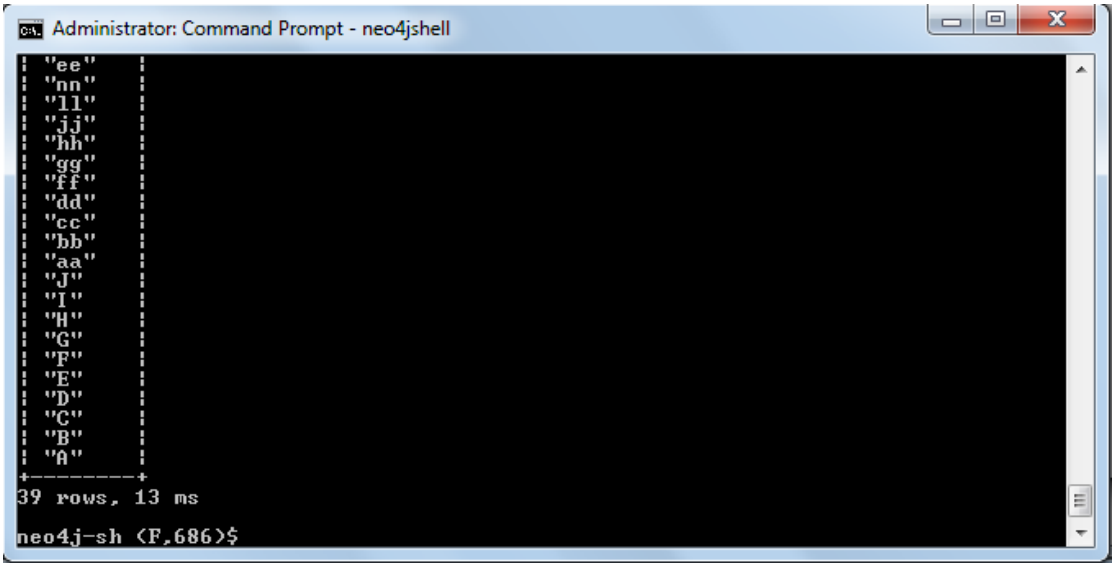


Figure 5.10 Retrieval times for 100 users for S1

The third query is to find the lead actors of favourite movie of Esha's friend. The Neo4j query for the same is:

- start n=node(192) match n-[:friends]->x-[:fav_mov]->y-[:acted_by]->z return z.name

Retrieval times for the above query as calculated by Neo4j for hundred and five hundred users are shown in Figure 5.11 and Figure 5.12 respectively.

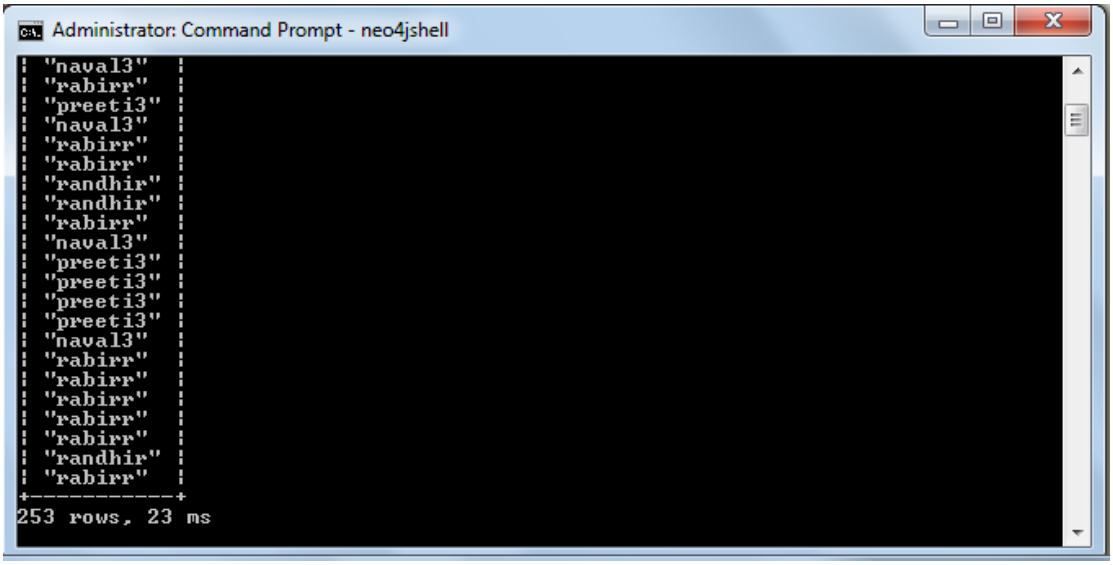


Figure 5.11 Retrieval time for 500 users for S2

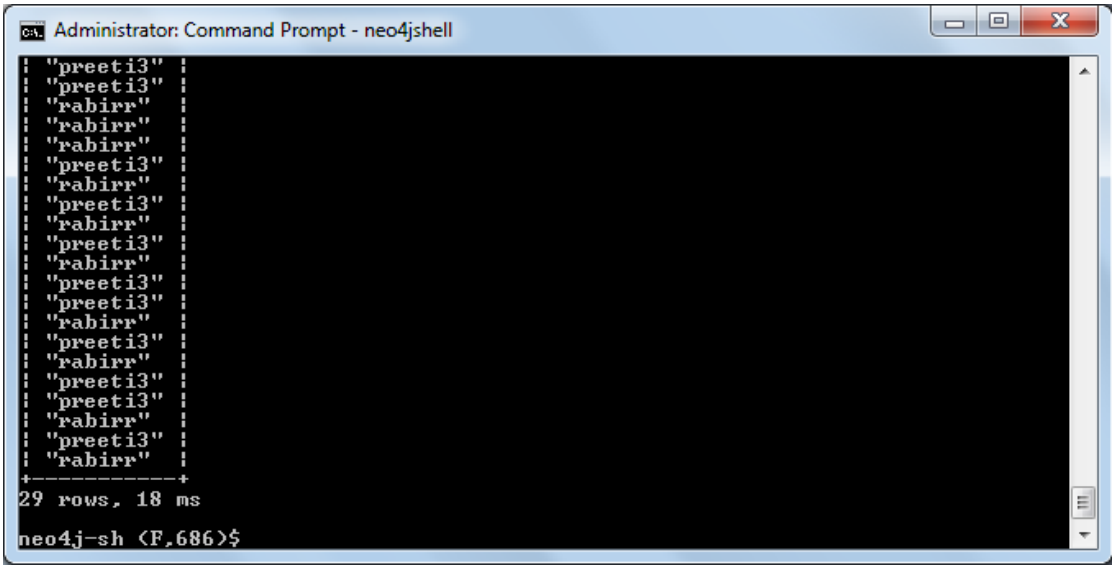


Figure 5.12 Retrieval time for 100 users

Step 5:

The next step is to query Neo4j to see if it gives the search result as expected by the user. The graph generated for our data model showing the subject , predicate and objects (triples) is as given in Figure5.13 and Figure 5.14

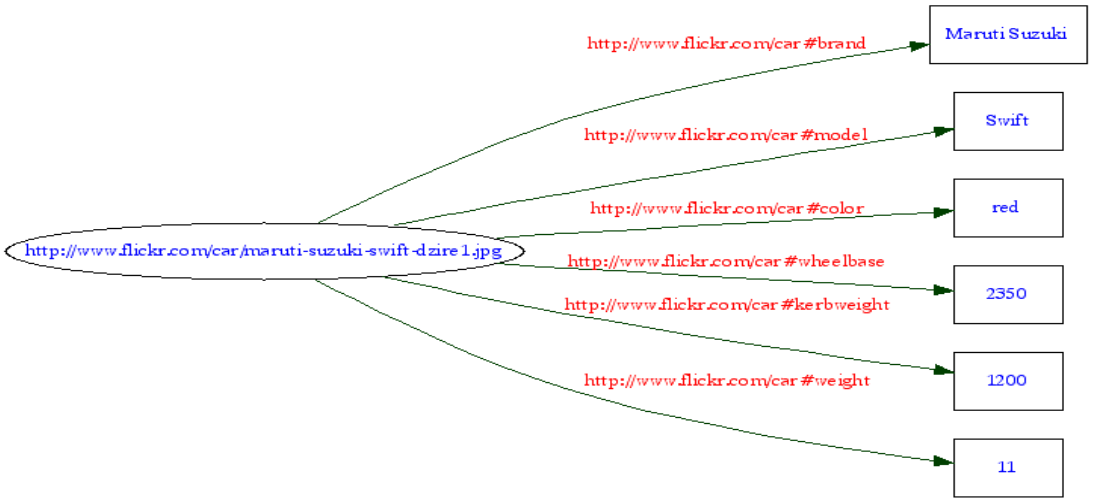


Figure 5.13 Graph Database Representation For triple Of Link1

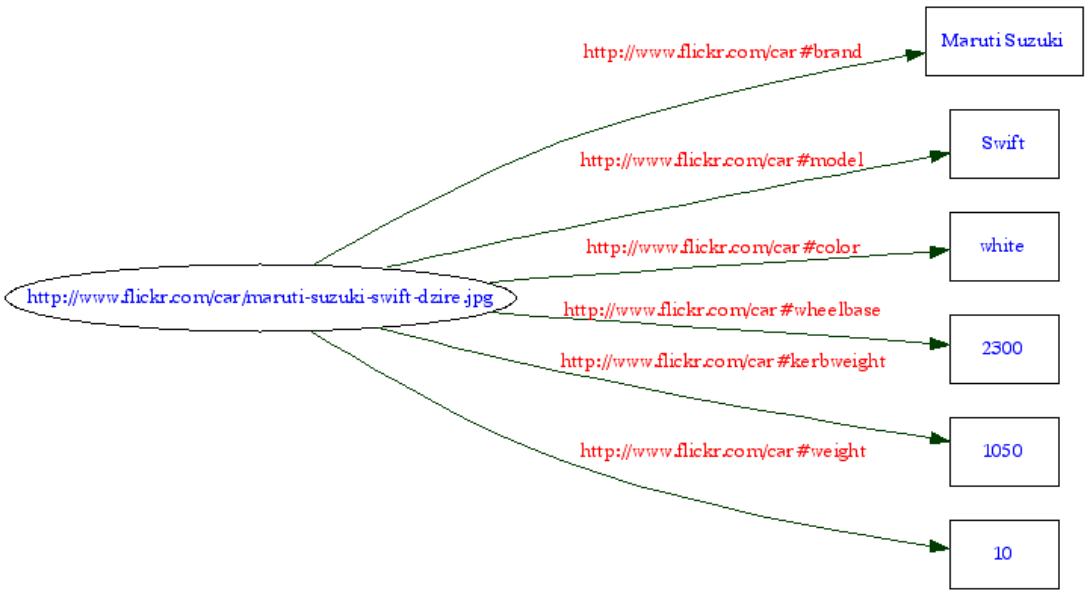


Figure 5.14 Graph Database Representation For triple Of Link2

Adding annotations to the document helps retrieve relevant information from the graph database. The next step is to retrieve the query from the database to check if it gives the results as expected by the user. So the query is to search the link for red swift car.

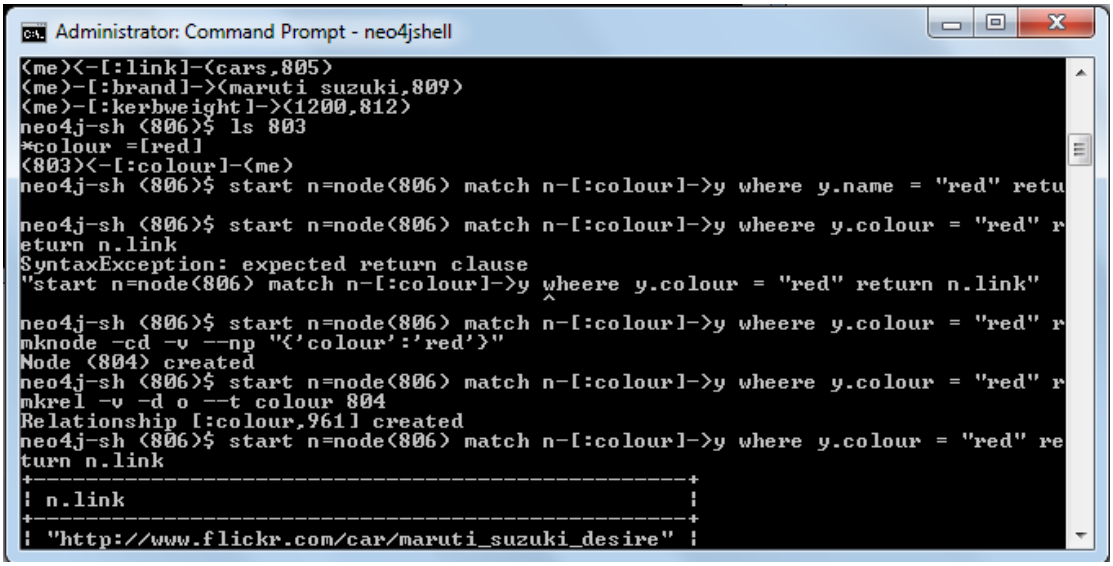


Figure 5.15 Implemantation Of Semantic Web In Graph Database

5.2 Implementation of MySQL

To implement relational databases, MySQL version 5.1.41 was used. The database was queried using PHP scripting language. Queries designed to analyze the performance difference between a relational database and a graph database are:

S0: Find all friends of Esha.

S1: Find the favorite movies of Esha's friends.

S2: Find the lead actors of favorite movies Esha's friends.

The first step is to connect to the MySQL server. Once the connection has been established to MySQL server, a new database is created.

Next step is to create tables within the database. Schema for relational database includes the following tables

1) User: user_id, user_name

2) Friends: user_id, friend_id

3) Fav_movies: user_id , movie_name

4) Actors: movie_name, actor_name

To retrieve the query results and note down the retrieval times of each query for hundred and five hundred nodes respectively. Unlike Neo4j, MySQL gives the

retrieval time in seconds and not milliseconds. To get the retrieval time in milliseconds, we need to explicitly do the coding. The database is queried using PHP scripting language. Retrieval times for the above query as calculated by MySQL for hundred and five hundred users are shown in Figure 5.16 and Figure 5.17 respectively.

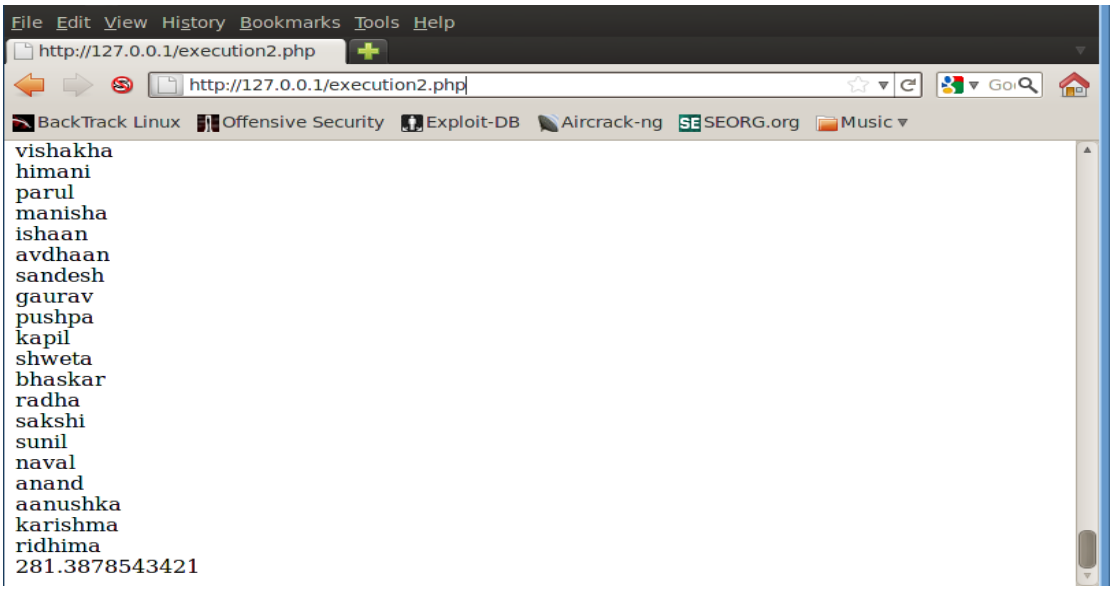


Figure 5.16 Retrieval time for 500 users for S0

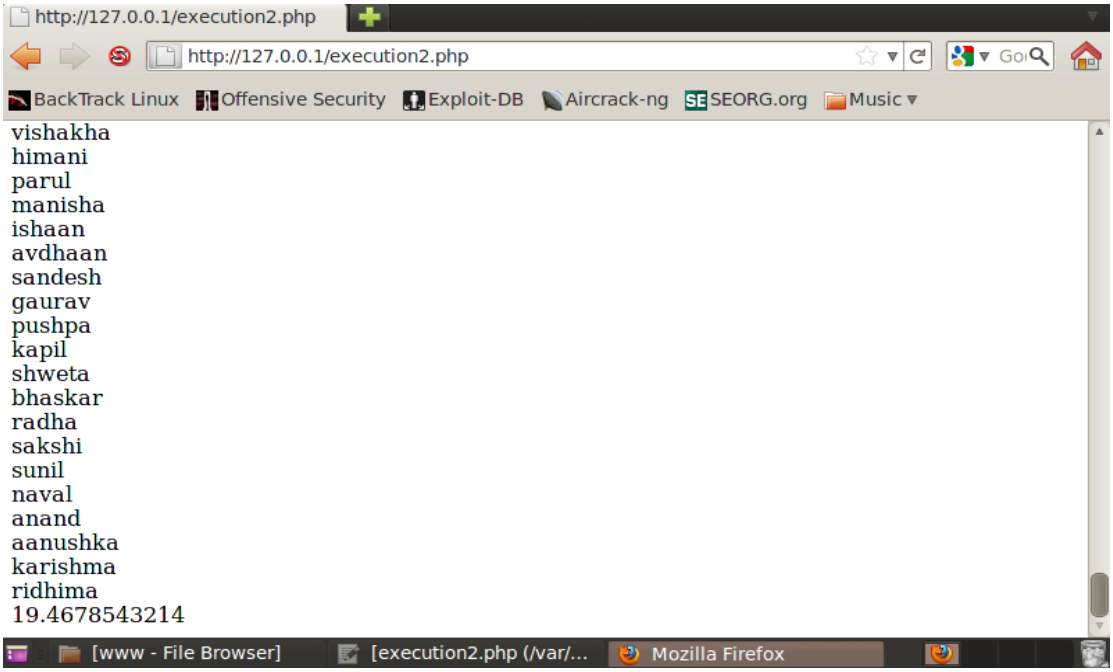


Figure 5.17 Retrieval times for 100 users for S0

The code is run for hundred and then for five hundred users. Each time the code is run, the cache has to be flushed in order to avoid any improvement in query retrieval time that may result due to caching of query results giving a wrong illusion that query is taking much less time.

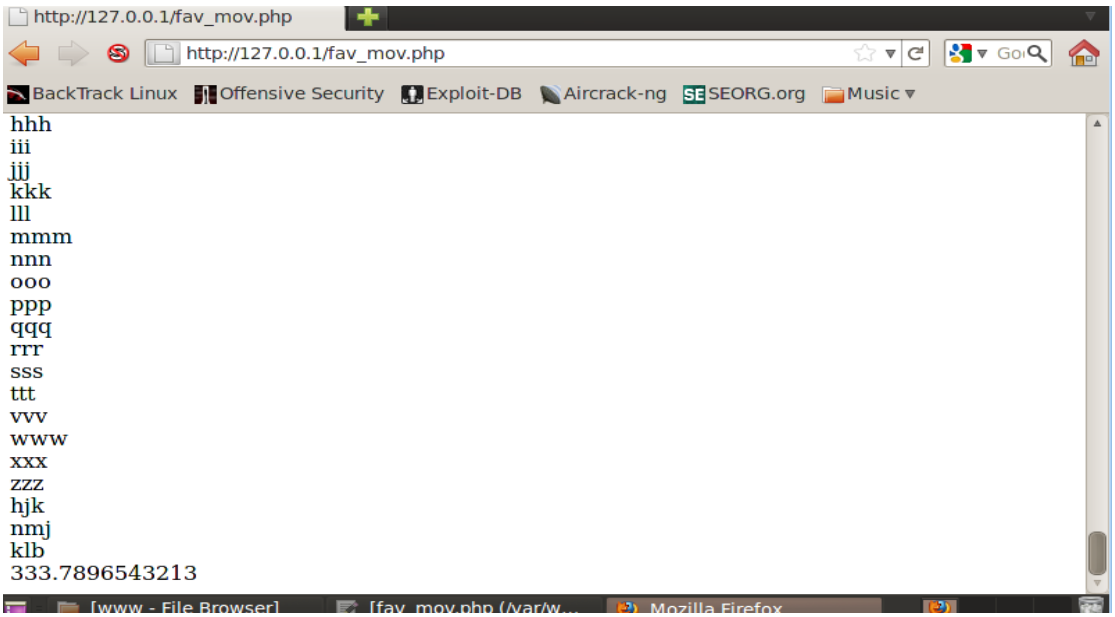


Figure 5.18 Retrieval time for 500 users For S1

Figure 5.19 Retrieval time for 100 S1

Step 5:
The next step is to query MySQL to see if it gives the search result as expected by the user.
This is how the same information is represented in relational database

Table 5.1 Relational Representaion Of Semantic Web

Pk	Link
1	http://www.flickr.com/maruti-suzuki-swift-desire1.jpg
2	http://www.flickr.com/maruti-suzuki-swift-desire.jpg

Table 5.2 Relational Representaion Of Semantic Web

pk	brand	model	color	wheelbase	kerbweight	Weight
1	Maruti suzuki	swift	red	2350	1200	11
2	Maruti Suzuku	Swift	White	1050	1100	8

5.3 Experimental Results

The results of the experiments have been tabulated in Table 5.3. It can be easily observed from the values retrieved (Table 5.3) that the retrieval times of Graph Databases is less than relational databases. This is because relational databases search all of the data to find the data that meets the search criteria. The larger the data set, the longer it takes to find matches, so when number of users increase from one hundred to five hundred, the retrieval time gets increased manifold. On the other hand graph database looks only at records that are directly connected to other records; it does not scan the entire graph to find the nodes that meet the search criteria. So, increasing number of nodes from one hundred to five hundred does not increase the retrieval time much as can be visualized from the graphs (Figure 5.20 and Figure 5.21).

Table 5.3 Query execution times in milliseconds

No_of_objects	MySQL:S0	Neo4j:S0	MySQL:S1	Neo4j:S1	MySQL:S2	Neo4j: S2
100	19.56	8	33	12.65	111.334	19.57
500	281.38	10	333.96	17	620.56	21

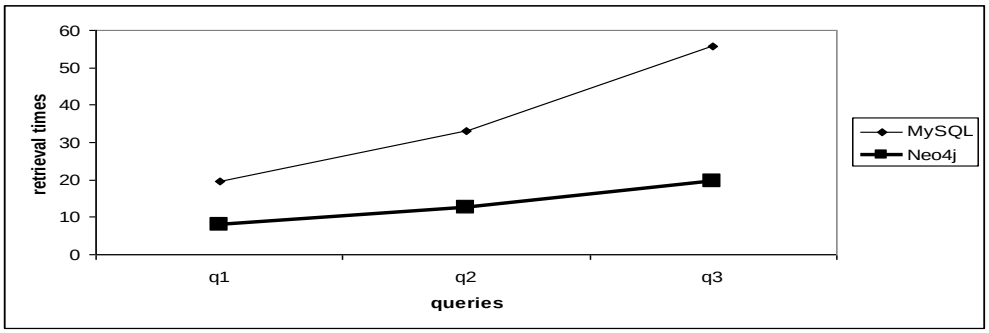


Figure 5.20 Retrieval times of queries by neo4j and mysql (100 objects)

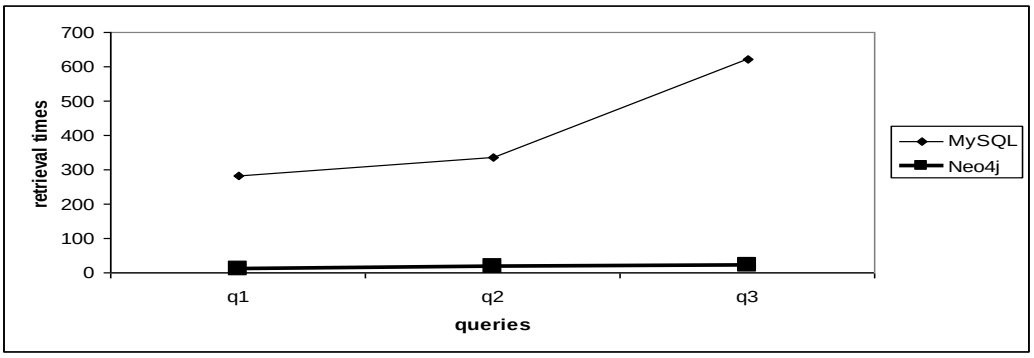


Figure 5.21 Retrieval times of queries by neo4j and mysql (500 objects)

As can be seen from table 5.1, Neo4j takes much lesser time for query retrieval as well as semantic search compared to relational database. This is because graph searches any element in $\log_b n$ time while relational database searches the same result in n^2 , n^3 depending upon the number of tables joined. As far as scalability is concerned ,even then Neo4j wins the battle.

5.4 Other Evaluation Parameters for relational database and graph database

The evaluation between MySQL and Neo4j is based upon different criteria [4]. These criteria are the pillars to decide which database should be adopted for implementation. The evaluation methodology designed to compare MySQL and Neo4j involves objective benchmarks and subjective comparisons based on system documentation and experience. The objective tests include processing speed based on a predefined

set of queries, scalability. Subjective tests include maturity/level of support, ease of programming, flexibility, and security.

A.Level Of Support/Maturity

Maturity refers to how well tested the system is. If a system has been tested more number of times it means it is more stable and more bugs have been found out. Maturity of a system is proportional to level of support. A more mature piece of software will have more people using it; more people talking about it on the web, in industry, and in academia and more people testing it in field. Relational Databases have been providing storage support for decades now. So they are more stable and mature. Both Oracle and MySQL have been providing extensive support for their commercial products. Relational databases have a unified language SQL. As SQL does not differ much between implementations, support for one implementation is applicable to others as well. Since Neo4j version 1.1 was released in February 2010, it is less stable and less mature. It lacks a unified language to interact with the database as query languages supported by Neo4j (SPARQL, Gremlin and Cypher Query) differ much in implementation. Support for one implementation is not applicable to all others. Neo4j is still growing and maturing and has not undergone the same rigorous performance testing as relational databases. Most of the support comes from its parent company's website www.neo4j.org and limited from outside of Neo4j site. While there is a small, vocal community, graph database user support pales in comparison to that of relational databases.

B.Security

MySQL has extensive multi user support. However Neo4j does not have any built in mechanisms for managing security restrictions and multiple users [11]. It presumes a trusted environment. Although there is Access Control List security mechanisms but even Access Control List management is handled at application layer. On the other hand, there is extensive support for ACL based security in MySQL.

C. Flexibility

Although relational databases are more mature and secure as compared to graph databases, but its schema is fixed, which makes it difficult to extend these databases [12] and less suitable to manage ad-hoc schemas that evolve over time.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

This thesis work concludes that in general, graph databases retrieve the results of the set of predefines query faster than relational databases. Not only this, graph databases are more flexible than relational databases as new relationships can be added to graph databases without the need to restructure the schema again . Graph databases are more scalable as well as increasing number of nodes from one hundred to five hundred does not increase the retrieval time by much as can be visualized from the Fig 5.20 and Fig 5.21). With such a difference in the query retrieval time of MySQL and Neo4j, Neo4j can be used for commercial purposes like website link structures and social networking.

Graph Database is a very powerful tool to annotate resources and create data models for the repositories of the different types of resources. The predefined tags used in Neo4j capture all the targeted information about the resources and put ahead a way to make all the searches over the defined repository more relevant and accurate in the most time efficient way as compared to relational databases.

6.2 Future Scope

Although Neo4j is far better than relational databases as far as query retrieval time and scalability is concerned. But it is still in the development phase and less mature. So there is a scope to make it more mature by performing rigorous tests on it. Currently Neo4j does not have any built in mechanisms for managing security restrictions and multiple users. Work can be extended in this aspect to make Neo4j more secure and easily available in multi user environment .

References

[1] Tony Marston, "The Relational Data Model, Normalisation and effective Database Design", Internet: http://www.tonymarston.net/php-mysql/database-design.html, 12th August 2005

[2] Paul Litwin, "Fundamentals of Relational Database Design", Internet: http://www.deeptraining.com/litwin/dbdesign/FundamentalsOfRelationalDatabaseDesign.aspx, [November 2011]

[3] Marko A. Rodriguez, July 2007, "Problem-Solving using Graph Traversals", AT&Ti Technical Talk - Glendale, California.

[4] Marko A. Rodriguez, April 8, 2009, "Graph Databases and the Future of Large-Scale Knowledge Management", Los Alamos National Laboratory.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, , 2008: "Bigtable: A distributed storage system for structured data". *ACM Trans. Comput. Syst.*, 26(2):1–26.

[6] G. DeCandia, D. Hastorun, M. Jampani,G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, 2007, "Dynamo: amazon's highly available key-value store". *SIGOPS Oper. Syst. Rev.*, 41(6):205–220,

[7] A. Lakshman., "Cassandra - a structured storage system on a p2p network", Internet: http: //www.facebook.com/note.php?note_id=24413138919, 2008.

[8] Charles Bachman, 1965. "Integrated Data Store." in: *DPMA Quarterly*.

[9] Codd, E. F., 1971 : "Normalized Data Base Structure: A Brief Tutorial", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, San Diego, available from ACM, New York.

[10] Codd, E. F, 1971: "A Data Base Sublanguage Founded on the Relational Calculus" Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, San Diego, available from ACM, New York.

[11] Codd, E. F., May 24-25, 1971: "Further Normalization of the Data Base Relational Model", Courant Computer Science Symposia 6,New York City, Prentice-Hall.

[12] Fausto Rabitti, Ellsa Bertino, Won Kim, And Darrell Woelk , 1994, "A Model of Authorization for Next-Generation Database Systems",Microelectronics and Computer Technology Corporation.

[13] A. S. Maiya and T. Y. Berger-Wolf, 2010, "Sampling community structure" . *www*, pages 701–710, New York, NY, USA,. ACM.

[14] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, 2008," Statistical properties of community structure in large social and information networks".. *WWW*, pages 695–704, New York, NY, USA, . ACM

.

[15] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, 2006,"Group formation in large social networks: membership, growth, and evolution" ,*SIGKDD*, pages 44–54, New York, USA,. ACM.

[16] H. Kashima and N. Abe,2007 " A parameterized probabilistic model of network evolution for supervised link prediction". *ICDM*.

[17] D. Liben-Nowell and J. M. Kleinberg, 2007,"The link-prediction problem for social networks" *JASIST*, 58(7):1019–1031.

[18] B. Taskar, M. fai Wong, P. Abbeel, and D. Koller, 2003 "Link prediction in relational data". In in Neural Information Processing Systems.

[19] A. Anagnostopoulos, R. Kumar, and M. Mahdian, 2008 “ Influence and correlation in social networks”. *SIGKDD*, pages 7–15, New York, NY, USA,. ACM.

[20] J. Tang, J. Sun, C. Wang, and Z. Yang,2009 “Social influence analysis in large-scale networks”, *SIGKDD*, pages 807–816, New York, NY, USA, ACM

[21] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proc. WSDM*, pages 241–250, New York, NY, USA, 2010. ACM.

[22] Roussopoulos, N., Mylopoulos, J., September 1975, "Using Semantic Networks for Database Management", Proceedings of the First International Conference on Very Large Databases, , 144-172.

[23] G.M. Kuper and M.Y. Vardi, April 1984,"A New Approach to Database Logic", ACM Int. Symp. on PODS”, Waterloo,

[24] HS Kunii , 1987,”DBMS with graph data model for knowledge handling”, ACM '87 Proceedings of the Fall Joint Computer Conference on Exploring technology.

[25] Chrrstophe Lbcluse, Phllrppe Rwhard and Fernando Velez GIP Altcur, 1988 “O2, an object-oriented data model”, SIGMOD '88 Proceedings of the 1988 ACM SIGMOD international conference on Management of data.

[26] Amann, Bernd, Scholl, Michel,1992 "Gram: A Graph Data Model and Query Language", Proceedings of the 1992 European Conference on Hypertext, ACM.

[27] Jan Hidders, Jan Paredaens , 1993, “GOAL, A Graph-Based Object and Association Language”. CISM - Advances in Database Systems 1993: 247-265

[28] D. Mukhopadhya, A. Banik, S. Mukherjee, and J. Bhattacharya, “A Domain Specific Ontology Based Semantic Web Search Engine”, Arxiv preprint, arXiv, 2011, Available: arxiv.org.

[29] Neo4j. Home. http://neo4j.org, 2012.

[30] D. Dominguez-Sal, P. Urb´on-Bayes, A. Gim´enez-Va˜n´o, S. G´omez Villamor, N. Mart´ınez-Baz´an, and J.L. Larriba-Pey, 2010"Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark", Proceeding WAIM'10 Proceedings of the 2010 international conference on Web-age information management.

[31] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, Dawn Wilkins,2010, "A Comparison of a Graph Database and a Relational Database", ACM Southeast Regional Conference.

[32] Database Trends And Applications, Internet: http://www.dbta.com/Articles/Columns/Notes-on-NoSQL/Graph-Databases-and-the-Value-They-Provide-74544.aspx,2012

[33] Chang, F., Dean, J., Ghemawat, 2008."Bigtable: A distributed storage system for structured data". ACM Trans. Comput. Syst. 26(2)

[34] Mart´ınez-Bazan, N., Munt´es-Mulero, V., G´omez-Villamor,2007,"Dex: high performance exploration on large graphs for information retrieval". In: CIKM pp.573–582.

[35]Jena-RDF. Jena documentation. Internet: http://jena.sourceforge.net/documentation.html , 2010.

[36] Sesame. Open RDF website, Internet: http://www.openrdf.org, 2010.

[37] Neo4j Blog, Internet :http://blog.neo4j.org/2009/04/current-database-debate-and-graph.html,2010.

[38] Neo4jmanual, Internet: http://docs.neo4j.org/chunked/stable/graphdb-neo4j-nodes.html.2010

[39] Neo4jmanual, Internet: http://docs.neo4j.org/chunked/stable/graphdb-neo4j-relationships.html

[40] Neo4jmanual, Internet: http://docs.neo4j.org/chunked/stable/query-start.html.

[41] Enhancing Image Search accuracy Using Semantic Annotation, (ME thesis) Satyavir.

List of Publications

Published

iv. [1] Charu Tyagi, Shalini Batra. "Comparative Analysis of Relational And Graph Databases" **International Journal of Soft Computing and Engineering (IJSCE)** Vol. 10, Issue. 10, May 2012 (Impact Factor: 0.52).