People Network - Knowledge Graph Using Neo4j

Karthik Vegi
Indiana University Bloomington
College Mall Apartments
Bloomington, Indiana 47401
kvegi@iu.com

Chaitanya Sagar Pullabhotla
Indiana University Bloomington
Woodbridge Apartments
Bloomington, Indiana 47408
cpullabh@umail.iu.edu

ABSTRACT

A graph database is a graphical representation of data points and the various relationships between them. This form of structure helps retrieve data faster using semantic queries. A knowledge graph is an implementation of the graph database which has found widespread applications in data management, social networks and other areas. In this project we try to build a network of people using Neo4j to understand the intricacies of using a graph database and the structure of a knowledge graph.

KEYWORDS

knowledge graph, RDF, data semantics, social network, Neo4j, graph database

1 INTRODUCTION

There are many popular graph databases which have been used in various applications across industries. Neo4j, developed by Neo4j, Inc. is a popular, open source graph database management system which has been used for building data management systems, social networks and fraud detection systems [5].

Neo4j uses graphs called Labeled Property Graphs to represent data in the form of Nodes and the relationships between the data points as edges [2]. This is a common representation of data in almost all graph databases making them more informative than traditional relational databases, where relationships can only be retrieved using complex queries.

Traditionally, most of the graph databases used RDF to represent connected data [5]. The main difference between the Labeled Property Graph and RDF is that the nodes and edges in Labeled Property Graphs have their own attribute values, giving the graph a more compact structure over the RDF model as shown in Figure 1. While the RDF model was developed to make data exchange easier, the Labeled Property Graph was developed to store the data efficiently i order to retrieve it much faster [2].

The other unique featue of Neo4j is the use of a new query language developed for faster graph traversal and data retrieval called Cypher [1]. While SPARQL queries are used to extract data from RDF models, Cypher quries are used to enter, update and retrieve data in Labeled Property Graphs. It borrows its basic structure from SQL and the pattern matching expressions from SPARQL [1]. The openCypher initiative by Neo4j aims to extend the use of Cypher to other graph databases [1]. Figure 2 shows the data representation in a graph database.

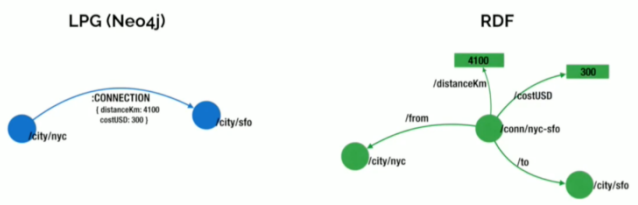

Figure 1: Labeled Property Graph

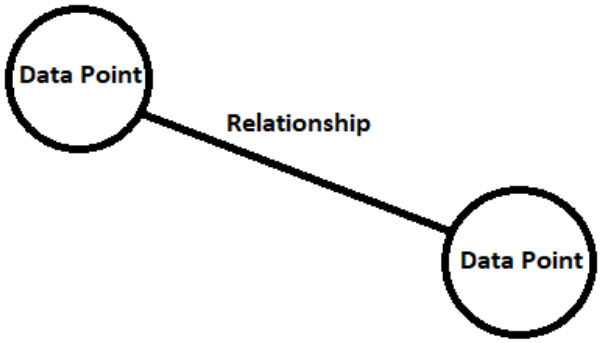

Figure 2: Data Structure: Nodes and Edges

2 PEOPLE NETWORK

A social network is an online platform that is used by the people to build and maintain social relations with their family, friends, colleagues or with people who share similar interests or activities [6]. Social networking sites are internet based web applications where the content is generated by the people who use the website [6]. They allow users to share ideas, photos, videos, posts to inform people about their online or real world activities [6]. People Network is a small social networking website built on top of a knowledge graph using Neo4j graph database.

In the application developed further, we have stored the data in the Neo4j database in the following format:
Nodes

- People
- Places

Edges

- Birthplace
- Friendship

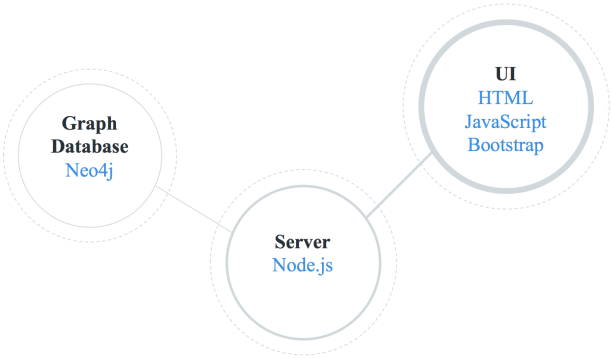

Figure 3: Application Stack

The application collects the names of various people and places and represents them in the form of nodes. It also establishes the relationships of people with other people by obtaining the names of people a person is friends with. It establishes the relationships of people with places by obtaining the name of a person's birthplace.

3 APPLICATION STACK

The following software was used as a part of the application stack:

- Graph Database: Neo4j
- Server: Node.js
- User Interface: HTML, CSS
- Dynamic Content: Javascript, JQuery
- Responsive Web Design: Bootstrap

Figure 3 shows the application stack.

3.1 Graph Database

Graph databases are built ground up to manage relationships. Graph databases embrace relationships as a core aspect of the data model which makes it easier to store, load, process and retrieve connections with great speed and efficiency [3]. The main difference between other databases and graph databases is that the other databases compute joins on after querying which takes up a lot of time where as the graph databases stores the connections in a very efficient way which makes the query on the fly very fast [3].

Graph databases also scale very well and can manage millions of connections with ease. Complex queries are also run efficiently on a graph database. Another advantage with graph databases is that it can be distributed over a cluster which makes it agile and blazingly fast [3].

3.2 Node.js

Node.js is a server side platform built on top of Google Chrome Javascript engine [4]. It is an open source server framework and uses asynchronous programming, in the sense it doesn't wait for the server after it issues a command and it immediately waits for the next command. It is event driven and designed to build scalable network applications [4]. Since no locks are employed, the users need not worry about the dead-locking process which makes it easier to build scalable applications [4].

The Node.js application integrates with the graph database Neo4j in the backend. The application uses Neo4j driver to connect to the running instance designed for Node.js which will connect, add, remove, and modify the graph connections in the background. The UI events trigger the search queries on the knowledge graph.

3.3 User Interface

The user interface has been designed using HTML, CSS for the layout and look. JavaScript with JQuery was used to design the user interactions and fire events. Twitter Bootstrap was used to make the web page responsive.

3.4 Node.js Modules

Node.js modules are application libraries that are designed to perform a particular task or events related to a particular task. The following node modules have been used in the applications:

- Body Parser: to parse the form data from the UI elements and pass it back and forth to the Neo4j database
- Express: Web framework to handle routes, which will link the pages to the corresponding routes
- EJS: Template engine to manage dynamic variables in HTML and Javascript
- Morgan: loggers to log the events, error messaging and other important messages
- Neo4j Driver: to communicate with the running neo4j instance in the background

3.5 Neo4j Driver

Neo4j driver for Node.js is required for the application to communicate with the neo4j instance. The Neo4j driver uses Bolt protocol to communicate with the database instance. The driver provides a session object which is used to fire the query and fetch the required data. Figure 4 shows the application stack.

4 APPLICATION DEPLOYMENT

The application set up and deployment can be done by executing the following steps:

(1) Download and install the Neo4j graph database.
(2) Start the database instance by firing up the localhost on port 7474
(3) Download and install Node.js server
(4) Download and unzip the code folder *people-network*
(5) Go into the code and install the node dependencies by using the command *npm install* which installs all the packages mentioned in package.json.
(6) Fire up node.js by using the command *node app* from inside the code directory which will start the server on port 3000.

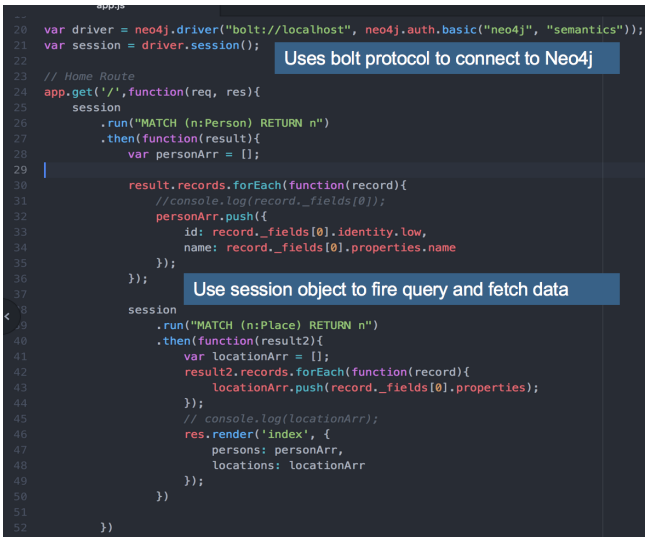

Figure 4: Using Neo4j driver

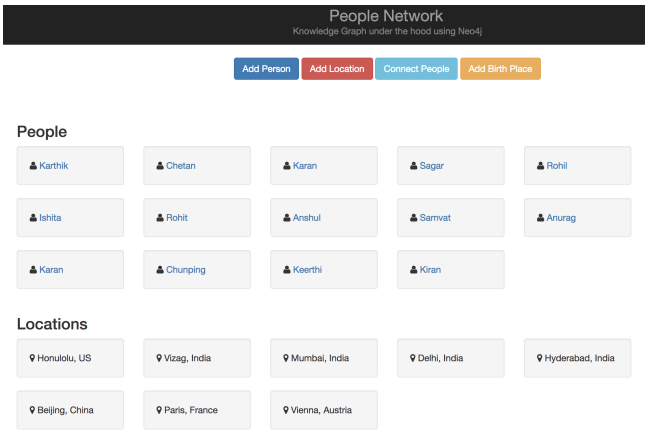

Figure 5: People Network Home Page

5 APPLICATION OVERVIEW

The following sections describe the layout and functionality of the application:

5.1 Home Page

The application's home page lists the people and locations on the network. Now you can see the home page of the application. Figure 5 shows the home page.

New nodes can be added from the home page using the different add options available. New relationships can also be established between the nodes using *Connect People* functionality.

5.2 Person Page

Clicking on the person will redirect to the person page. The url is routed using the person id which is not a part of the attribute

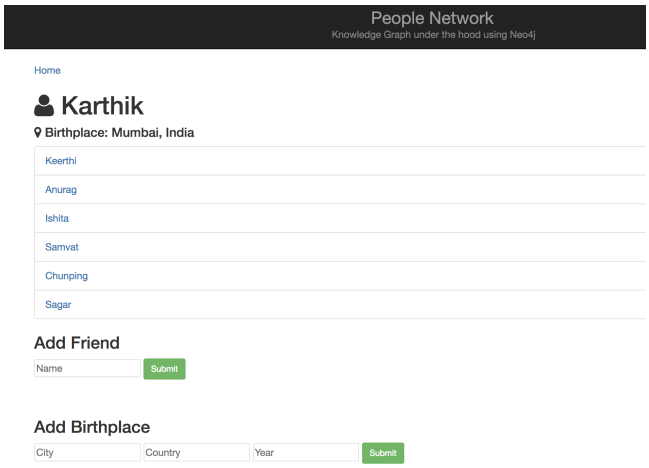

Figure 6: People Network Person page

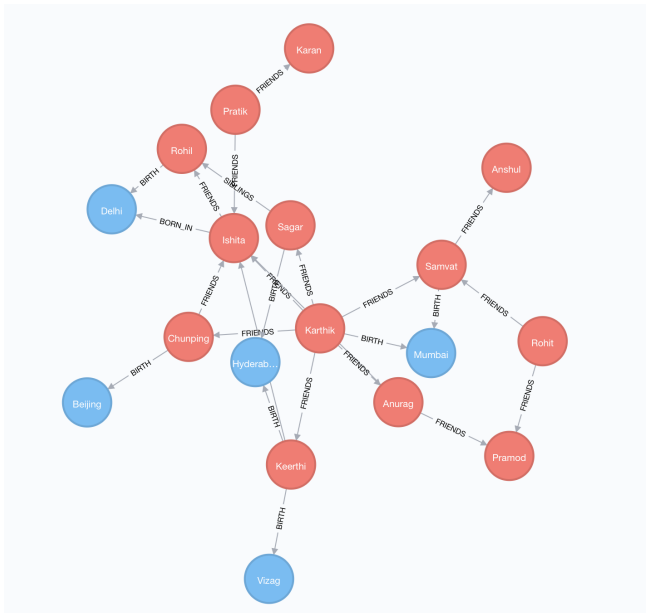

Figure 7: Knowledge Graph

list. The id is used internally to identify and manage nodes. The person page will show his birthplace if available and then lists all his connections as a list. Clicking on each of these connections will redirect to their respective pages. A new connection can be added to the person using the add friend functionality. A birthplace can also be added from the person page which adds a new attribute with the location as the value. Figure 6 shows the person page.

5.3 Knowledge Graph

Neo4j handles the knowledge graph under the hood by managing the nodes and the connections between the nodes. Figure 7 shows the knowledge graph under the hood.

The knowledge graph shows the nodes of people in orange and the nodes of locations in blue. The connections are established using edges and the edges are marked with the relationships.

6 FUTURE SCOPE

The application is limited in the UI elements which run complex queries under the hood. Keeping the time and resources in view, the application has limited functionality. However, this can be further improved by adding new UI elements to add different types of nodes and establish different kinds of relationships between them. Also the application can also be connected to a social networking service like Facebook to extract an existing knowledge graph alongside creating new nodes and relationships.

7 CONCLUSION

Thus we see how a perfectly working Social Network of people can be easily developed using Neo4j. It helped us visually interpret the data clearly and ease with which Cypher queries could be used to enter, modify and delete data. More fuctionalities can be added to the application to make it more sophisticated and we have seen how to do that using the various technologies that we have discussed above. Overall, Neo4j is a very efficient graph database and it would be interesting to explore other areas where such a graphical representation of data could prove useful.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Ying Ding and the teaching assistant Yi Bu for their support and suggestions throughout.

A WORK BREAKDOWN

Project Identification: Karthik Vegi, Chaitanya Sagar
Requirement Gathering: Karthik Vegi, Chaitanya Sagar
Learning Neo4j and Implementation Chaitanya Sagar
Learning Neo4j and Implementation Karthik Vegi
Coding and Designing Home Page: Karthik Vegi.
Coding and Designing Person Page: Chaitanya Sagar
Writing the project report: Karthik Vegi, Chaitanya Sagar

REFERENCES

[1] Neo4j. 2014. Cypher Query Language. Webpage. (2014). https://neo4j.com/developer/cypher-query-language/
[2] Neo4j. 2016. Triple Store Vs Labeled Property Graph. Webpage. (2016). https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/
[3] neo4j. 2017. Graph Database. Webpage. (2017). https://neo4j.com/developer/graph-database/
[4] Node.js. 2015. About Node.js. Webpage. (2015). https://nodejs.org/en/about/
[5] Wiki. 2005. About Graph Databases. Webpage. (2005). https://en.wikipedia.org/wiki/Graph_database
[6] Wikipedia. 2003. Social Networking Service. Webpage. (2003). https://en.wikipedia.org/wiki/Social_networking_service