GeoSpatial Intelligence on a Graph

Tharik Kanaka[1], PPG Dinesh Asanka[2]
[1]Sri Lanka Institute of Information Technology,
Colombo, Sri Lanka
[2] Pearson Lanka (Pvt) Ltd, Colombo, Sri Lanka
tharik.kanaka@gmail.com, dineshasanka@gmail.com

ABSTRACT: *There is a gap when representing detailed information related to public on maps due to limitations of underlying data sources. The purpose of this research is to scan the research output on Geographical Information Systems (GIS) and relational databases, then develop an implementation by combining these two technologies GIS and graph databases in order to fill the identified research gap. The outcome of this implementation is a central Geospatial intelligence on graph which delivers benefits in the context of multiple data source integration and open sources.*

Keywords: GIS, Graph Databases, NoSQL, Geospatial

Received: 7 January 2015, Revised 12 February 2015, Accepted 2 March 2015

1. Introduction

The past few decades with the usage of computer systems each and every organisation possesses piles of data and information. Most of them are stored in relational databases as relational database would be helpful for users to perform their data needs. Apart from the day to day processing, they have to perform complex queries on databases for business intelligence purposes. This has been achieved by integrating data in different databases and do some analysis on them to improve their business qualitatively and quantitatively. This is achieved by business intelligence techniques such as data warehousing, data mining, what-if analysis etc. Later with the usage of Internet, people began to interact with search engines, emails and social web sites [1].

The increment of internet usage leads to large amount of complex data which cannot be manipulated with ordinary database management system. As a result of that many concepts such as Big Data came to picture.

Apart from internet users, now data is exponentially generated by devices which are connected to internet than users [2]. This trend is known as “Internet of things”. As a result of that rapid growth can be expected in future. In order to manipulate big data, the traditional relational database management systems changed to a new concept called “NoSQL” (NoSQL stands for Not only SQL) which means beyond the traditional relational databases, there are more ways to manipulate those Big Data. The research paper published by Google in 2004 on the topic “Map Reduce: Simplified Data Processing on Large” Clusters [3], which can be identified as the starting point of Big Data concept since it introduced a method called Map Reduce to process and store large set of data with a parallel, distributed algorithm on a cluster. Various researchers and organisations began to search on this topic. Lately as an outcome there are several types of NoSQL data models namely Key-values Stores, Column Family Stores, Document Databases and Graph Databases. So Graph databases are one of the NoSQL databases which will be considered in this research [4].

In addition, Geographic Information System (GIS) became popular as it has the capacities of presenting all types of geographic data. GIS allows to capture, store, manipulate, analyse manage and present data. In early days, all these data were stored in flat files then lately it was developed to store in relational databases with the help of spatial indexes. Now it's time to implement GIS with Big data. After conducting a literature review on NoSQL and GIS, it is clear that storing GIS data on NoSQL database enabled to perform complex queries.

2. GIS and Graph Integration

The present day's studies of graphs networks were started in 1736 as a result of analysis done by the Swiss mathematician called Leonhard Euler. He considered a problem that was a curiosity among the citizens of a city called "Koenigsberg" in former East Prussia. It was about seven bridges that cross a river and two islands, and problem was that is it possible to walk across each bridge exactly in one trip? [5]. Euler solved the problem by considering the bridges and land masses as parts of a graph, an abstract set of junction objects and edge objects which are bridges. He came up with an easily understandable theorem that starts his observation that the junction where you start a tour must be connected to an odd number of edges. According to Euler's observation and analysis it is very clear that we abstract a geographic system to its underlying graph [6]. Present days this abstraction will be done by GIS software widely available. This underlying graph enables you to perform network analysis such as finding the shortest route etc. Network trace supports many algorithms from graph theory with GIS software implementation. When map interactions such as routing are carried out on a network in GIS software, actually behind the scene the interaction is done with its underlying graph. GIS software able manages bidirectional links between network features and elements in a graph so that you can naturally interact with network features to solve useful problems.

There are advantages when representing large amount of data with so many relationships on the graph databases over the relational databases which are widely used. Although, there are advantages on representing GIS data on a relational database system, representing large amount of data will be a challenge in relational database system. If there is a possible way to represent those data on a graph database there are a lot significant advantages could gain of graph theories. Actually there are few graph databases which are being used in the industry and also graph databases such as Neo4j which supports representation of GIS data. In Neo4j, there is a special library of utilities called Neo4j Spatial that facilitates the enabling of spatial operations on data. It can be used for so many GIS and for so many enhancements such as adding Geospatial intelligence to existing graph database models and as well as to retrieve results by executing more complex spatial queries on the data model.

In present days, geometry information has taken formation by providing map features with social information. One example is "*Foursquare*" which provides search capabilities of social network and mapping information where social connections have been visited and the reviews [7]. When considering about social platforms such has Facebook, the core data representation has been moved to graph structure. The Facebook graph known as Unicorn has been clearly mentioned in the research paper Unicorn: A System for Searching the Social Graph [8].Facebook is not only social network has been moved to graph; even popular Twitter platform has implemented their structure on top a graph called FlockDB [9]. Even though these different social networks have their own graphs they have been looked for integration among themselves. The Open Graph protocol [10] is such result where it has defined set of standard syntaxes to integrate information of various sources such as websites, blogs and forum posts to social networks. Typical scenario is, sharing a personal blog post in Facebook where that particular blog post has been treated as a node and link that post to sharing person with a relationship. If others share or comment on the post, they will be represented as relationships for that particular node. Therefore, integration is a key feature in graphs for knowledge bases. Integrating GIS with a graph source will be a wide knowledge base which can be useful for geospatial intelligence.

3. Architecture

The design implementation is consist multiple modules such as database module which is a Neo4j database and API Interface which should integrated with database expose features and client interface. The Neo4j will be the central graph databasewhich provides cypher query interface as well as REST API [11], which can be used to integrate with third party applications

This API allows creating, reading, updating and deleting nodes and relationships on the database. Since API is a RESTful service interface, it is inter-operable can be easily access by any applications by using HTTP Protocols such as GET, PUT, POST, and DELETE. As per research intentions this default API cannot be exposed to third party due to several reasons such as:

• API is designed to do database operations and not GIS operations.
• Granting access permissions to API is almost granting permissions on main database and difficult to maintain third party user roles.
• Cannot control the API user load and performance by enabling throttling mechanisms.

Therefore, it is required implementing a separate API which wraps Neo4J API operations in a manner to present third party as GIS API operations which make sense and also it should be able to manage operations, enhance security and able to monitor and control traffic on API usage.

As shown in the Figure 3.1: Implementation overall diagram the WSO2 API Manager is been placed in between Ne4j API and Client Application which runs on top of a browser. Server end is distributed where API Manager and Ne4j can be deployed to separate servers and still can communicate over HTTP protocol. There can multiple client applications communicate with WSO2 API Manager to get access to the Neo4j database. Security can be achieved by API manager with OAuth tokens where as API accessing authorisation privileges can be defined for OAuth tokens. Other than that API calling load and throttling can be managed at API Manager. Since HTTP is a lightweight widely used protocol around the world, the interoperability of this implementation is very high. Even any application written on any language and deployed on any platform can communicate with the back end by using HTTP protocol. In this implementation front end is an simple HTML web page with javascripts which executed on browser. At the browser, Geospatial information is retrieved from API Manager and Map related information is retrieved by OpenStreetMap with the support of Leaflet Mapping Library.

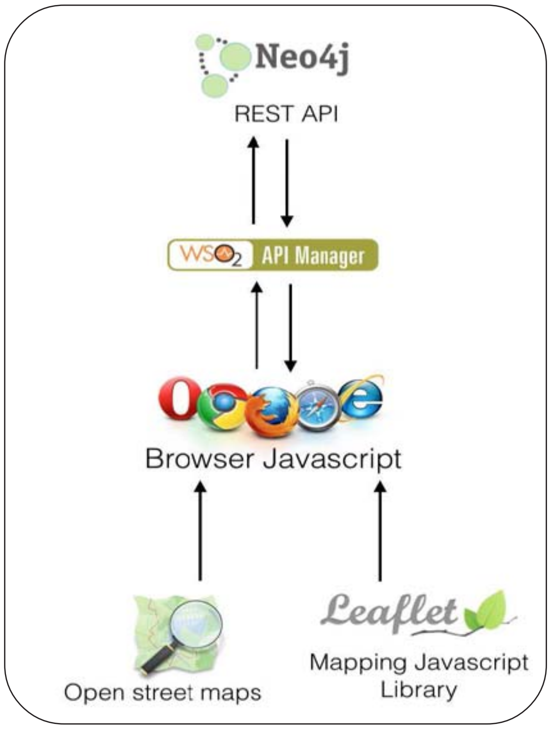

Figure 3.1. Implementation Overall Diagram

Another ability of this implementation is the option of deploying total solution on a cloud environment without hosting on premises. Practically there could be large amount of data when implementing graph database for this use case, it requires large amount space to store as well as large amount of resources to process those data and serve to end users. Initially data can be small and processing can be less but as time goes on these can be get increased exponentially. This situation can be managed by hosting implementation on a cloud environment. Neo4j database can be configured on a cloud SaaS (Software as a service) provider such as Graphendb which runs on hosted instances such as Amazon AWS or Heroku. End user has to consider only about what size of graph database they require and what is the bandwidth load for that. By deploying on cloud, there are multiple advantages such as scalability, automatic backups, and preconfigured plugins etc. A product like WSO2 API Manager can be hosted on WSO2 Cloud SaaS which also provides set of advantages as above. At the moment it has been launched as API Cloud. Capability of deploying on a cloud is a major advantage of this design to face and solve the scalability and performance issues which will leads to databases starts to grow and interacting users and third party applications get increased exponentially. Therefore, this implementation can be name cloud computing capable out of the box without doing tedious changes.

4. Results

When location nodes are retrieved from the API interface, the order of the nodes are not sorted properly. To verify the implementation query was executed to get results where restaurants with submarine and cinema halls in which Rio 2 is screened and fuel stations which are selling Octane 95. If unsorted data is displayed on the map the direction would be as follows where path is not optimised to shortest paths.

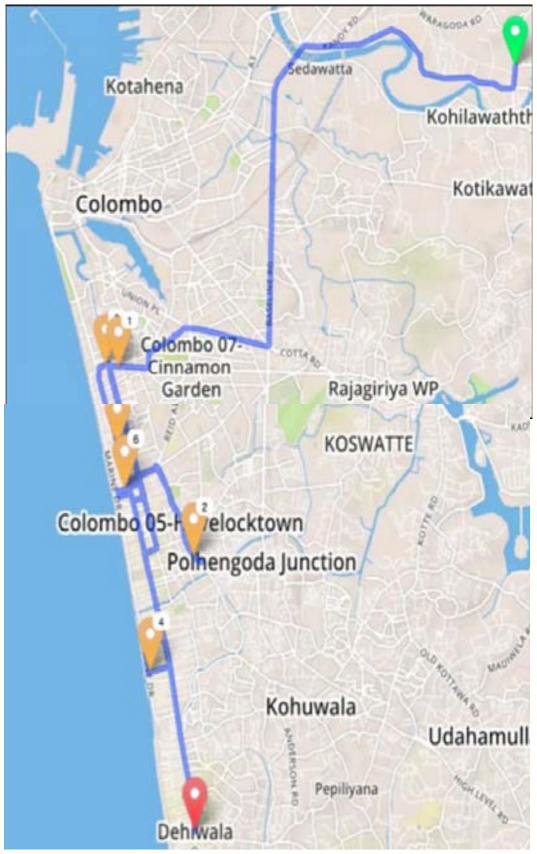

Figure 4.1. Non-optimised route

Above query result suggests go to Dinemore restaurant in Kollupitiya then suggests go to Laugh Fuel Station in Bambalapitiya and again come back to Liberty Cinema Hall in Kollupitiya then again go to Dinemore restaurant in Wellawata and come back to Fuel Station in Kollupitiya. After all, it suggests go to MC Cinema Hall in Bambalapitiya and then go to destination. This non optimised journey is 37.0 KM long. This can be optimised by sorting these points in a proper algorithm. For this research purpose an algorithm has been used to sort based on longitudinal and latitudinal positions of the location points. First new attribute called distance has been added every node which is initialised to 0. Then select first node and calculate distance for first node from other nodes and then update each ones distance accordingly. The distance between two points has been calculated by using following function. In the function longitude and latitude points of two locations as taken as parameters and convert those to radians. Then calculate point distance by trigonometry. Finally by considering Earths curvature actual distance will be calculated.

```
function calculateDistance(lat1, lon1, lat2, lon2, unit) {
var radlat1 = Math.PI * lat1/180
var radlat2 = Math.PI * lat2/180
var radlon1 = Math.PI * lon1/180
var radlon2 = Math.PI * lon2/180
var theta = lon1-lon2
var radtheta = Math.PI * theta/180
var dist = Math.sin(radlat1) * Math.sin(radlat2) + Math.cos(radlat1)
* Math.cos(radlat2) * Math.cos(radtheta);
dist = Math.acos(dist)
dist = dist * 180/Math.PI
dist = dist * 60 * 1.1515
if (unit==”K”) { dist = dist * 1.609344 }
if (unit==”N”) { dist = dist * 0.8684 }
return dist
}
```

Above function is been called as shown in following Figure 4.2: Sorting locations. Assume that there are 5 location nodes. First new distance attribute is added and initialised to 0. Then first node will be selected and distance from location node A will be calculated for all the location nodes. Then sort the list by distance attribute.

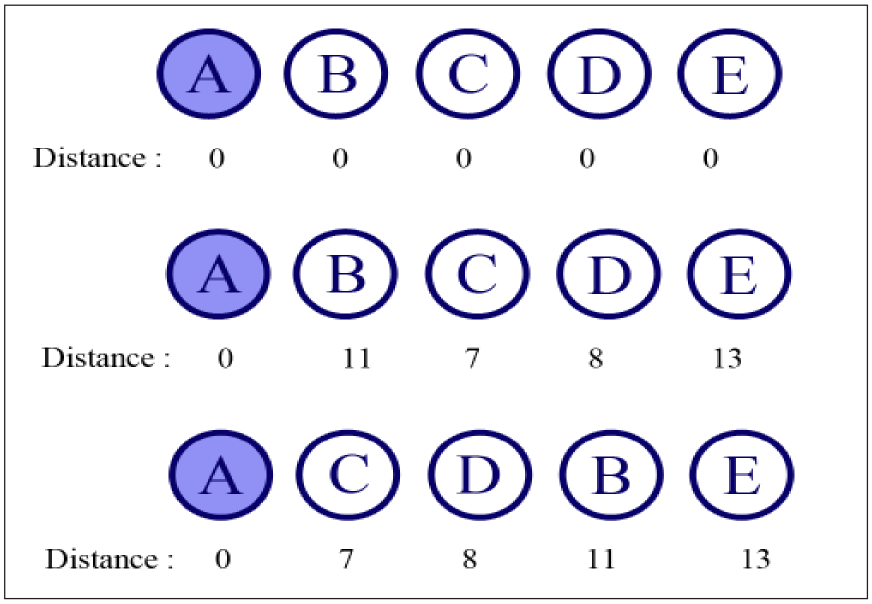

Figure 4.2. Sorting Locations

Above map route is based on graph query to show all restaurants which have Submarine food, Cinema Halls which shows “Rio 2” and Fuel Stations that have “Octane 95” Petrol. This query has been invoked through API Manager Application as a cypher query, as result there will be 6 nodes (2 restaurants, 2 Fuel Stations, 2 Cinema Halls). The travel route can be optimised by clicking on additional restaurants, fuel stations and cinema halls based on user preferences and optimised route will be generated on the map for remaining nodes accordingly. On the hand the directions has been displayed on left hand side and also total distance is distance and direction distances are shown.

For start and destination locations address lookup and geocoding is done and also it is capable of search of longitude and latitude points. This is just another scenario of demonstrating extensibility of platform on particular routing mechanism

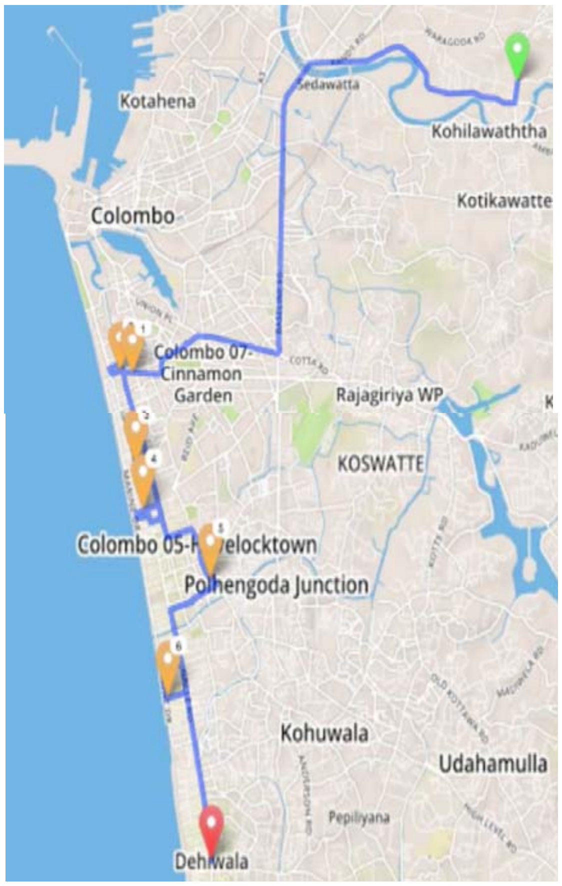

Figure 4.3. Optimised Route

likewise this implementation is capable of integrating other map routing services such as Google Maps. Another key feature is usage of open source products for interface which gives developers to freedom to use it and modify as required, in this case modifying to support graph is not capable if these are not open source products. The graph query can be constructed by dynamically by providing user controls such as drop down boxes where initial data about node label types, nodes, relationships can be loaded dynamically by calling API back end. In Figure 416: Filter User Controls shows hoe user can select filters and add multiple filters for search. Here user queries Fuel stations which have petrol of type Octane 95. JavaScript code can check values of combo boxes and generate cypher query.

5. Limitations and Future Work

The main limitation of the implementation is not having a distributed cluster of machines for graph database. Although the overall implementation is distributed where database end, API Manager and front end can be deployed on distributed environments by including features such as load balancing, redundancy but clustering database also deliver significant advantage when database grows and to support deep database mining and query data analytics on top of that.

In this research the geospatial intelligence data is captured in the graph database, by using API those data can be retrieved or updated for authenticated users using API Manager. On the hand according literature review the OpenStreetMap is also a open data collection which is been managed OpenStreetMap itself, but it also provide an API for to update data for OpenStreetMap users. Therefore this research implementation can be further improved by modifying API services with multiple actions to update both underlying graph database API and also OpenStreetMap API. For this purpose it requires to include OpenStreetMap credentials to service interface and also it requires message transformations to transform current message according OpenStreetMap specifications in order to communicate.

This work has laid the technical platform to develop a central knowledge base which could hold geospatial intelligence on a graph database, for further improvements anyone can continue the research work by collecting real data from a particular area and try to implement on the knowledge base. As an example data collection can be carried out on railway department to gain information about train schedules. When it comes to train schedules it is not just time, it can be more detail such as size of the train such as how many compartments? How much is the seat capacity? Also some details such type of the train weather its commuter train or intercity train with classes.

Since this is a centralised graph database, the size will get increased heavily as usage increase, thanks to the graph architecture performance will not be a huge problem but if it requires to cluster and perform distributed processing, the geospatial graph database can be deployed on some environment like a Apache Hadoop cluster which allows for the distributed processing of large data sets across clusters of computers.

5. Conclusion

For this research, the background study was carried out on the usage of GIS applications in industries and what type of technologies involve and significance of NoSQL databases and graph databases in order to develop a new implementation on integrating those technologies to come up with a central system with API facilities. All the geographical information and relations will be represented and stored in a graph database such as Neo4j graph. Then it presents the implementation carried out to fulfil the research aims and objectives. In the implementation, an API Manager application has been used to expose services in an efficient manner. Then a platform independent JavaScript base client application has been implemented to communicate with API Manager with REST HTTP protocol and retrieved data has been rendered on a map by using a map engine and mapping library. The final outcome of this research is implementation which can be used to store retrieve information about maps and other organisation information. This knowledge graph can be extended by plugging multiple sub graph which represents different set of domains. Also, this is capable of being connected with social platforms and increases the knowledge base. For an instance there can be branches of a restaurant in multiple cities which can be represented in a graph. Peoples who have visited those places and their reviews, feedbacks also can be connected to same graph. This information can be originated from various social platforms.

5. Acknowledgment

The research implementation is consist of multiple open source free software implementations which are developed, maintained and contributed by the community, This research would not have been possible without the existing projects which are supported by open source community and contributors. Those open source products lay platforms many researches.

References

[1] Internet Live Stats. Internet users. Available: http://www.internetlivestats.com/internet-users (Accessed: 23 March 2014).

[2] 12 Ahead. Google’s journey to becoming the operating system of everything. Available: http://www.12ahead.com/google%E2%80%99s-journeybecoming.

operating-system-everything (Accessed: 29 March 2014).

[3] Jeffrey Dean., Sanjay Ghemawat. (2004). MapReduce: Simplified Data Processing on Large Clusters.

[4] Digital Ocean. A Comparison Of NoSQL Database Management Systems And Models. Available: https://www.digitalocean.com/community/tutorials/acomparison-of-nosql-database-management-systems-and-models (Accessed: 24 May 2014).

[5] Esri. Networks and Graphs. http://resources.esri.com/help/9.3/arcgisengine/dotnet/e084da94-d4f7-4da7-86ed-7df684ff2144.htm (Accessed: 21 December 2014).

[6] Michael Curtiss., Iain Becker., Tudor Bosman., Sergey Doroshenko., Lucian Grijincu., Tom Jackson., Sandhya Kunnatur., Soren Lassen., Philip Pronin., Sriram Sankar., Guanghao Shen., Gintaras Woss., Chao Yang., Ning Zhang. Unicorn: A System for Searching the Social Graph. 2013.

[7] New York University. Foursquare Explained. Available : http://www.nyu.edu/content/dam/nyu/studentAffairs/images/Explained/foursquare.pdf (Accessed: 21 December 2014).

[8] Twitter (2010). Introducing FlockDB. Available: https://blog.twitter.com/2010/introducing-flockdb (Accessed: 01 June

2014).

[9] Open Web Foundation. The Open Graph protocol. Available: http://ogp.me/ (Accessed: 17 June 2014).

[10] Neo4j (2014).The Neo4j Manual - REST API. Available : http://neo4j.com/docs/stable/rest-api.html (Accessed : 06 July 2014).

[11] Ian Robinson, Graph Databases, 1 ed, O'Reilly Media, 2013.