The Modelling of Graph with Neo4j in Social Networks

MSc. Ibrahim Mete Cicek

Published online: 3 January 2020

Abstract

Technology is growing and getting a complex structure every day. In this case, errors occur in the systems, and the probability of failure increases. Line and a connection break in communication lines due to various reasons cause malfunctions in the systems. As a result, there may be huge irreparable losses. The main purpose of our article is to show and analyze the use of graphs in social networks, which are the sub-branch of topology that examines the relations between two points, which are used in all fields. We will do this using neo4j which is the most popular database in the world. Suggesting the correct graph model, algorithm, and database structure to the user and providing information on this subject will have a great contribution to prevent data and time losses caused by possible connection errors in human life, communication security, and real-case scenarios in social networks.

MSc. Ibrahim Mete Cicek
Department of Computer Engineering,
Faculty of Engineering,
Karabuk University
78050 Karabuk, Turkey
metecicek@karabuk.edu.tr
Lecturer: Prof. Dr. Ismail Rakip Karas

Keywords: *Graph Theory, Social Networks, Social Graph, Neo4j, NoSQL, Graph Database, Social Media*

Introduction

A graph consists of a non-empty set of vertices, and a set of edges that connect (pairs of) vertices. There are different types of graphs like undirected, directed, simple, multigraph, etc. There are many real-life scenarios where the graph data structure is used such as Neo4j.
Graph theory refers to the study of different types of graphs which are in the form of mathematical structures. These structures are used to make models of pairwise relations that exist between different objects. A graph is made up of vertices or nodes connected by edges or links.
The degree of a node in the graph is the total edges that are incident on the vertex and this defines how effective the importance of that node is.
Different properties of graphs are used in different social networks. Some of the properties of graphs that are used in the analysis of social network involve the measure of the path which can be defined as a sequence of vertices included by following connected edges across the network, or identification of a geodesic path which can be defined as the shortest path, in terms of several edges traversed that exists between a specific pair of vertices. Another property involves finding the mean geodesic distance that exists between a vertex and all other connected vertices to that vertex. This is also called closeness.
Other properties of the graph include kinship structure which is a processor the principle by which individual or groups of people are given different roles and categories and organized based

on these things, one other property is the distribution of structural properties such that these elements can be further analyzed and can help in gaining insight into the network in a deeper manner.

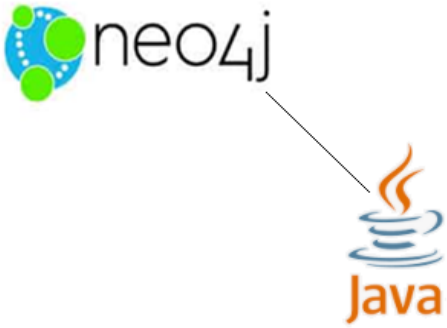

Figure 1 *The last letter of Neo4j represents the java programming language.*

Used Terms of Graph Theory in Social Networks

A social graph refers to the graph that depicts all the people you know and the people they know.
A digital graph refers to the graph that is a mapping of all online social relationships which are significant.
A social graph is nothing but a diagram that is used to illustrate the interconnections among people, groups, and organizations in a social network.
The term can also be used to describe an individual's social network. A social network on the other hand is constituted by several units or nodes that are connected by a specific and well-defined relationship.

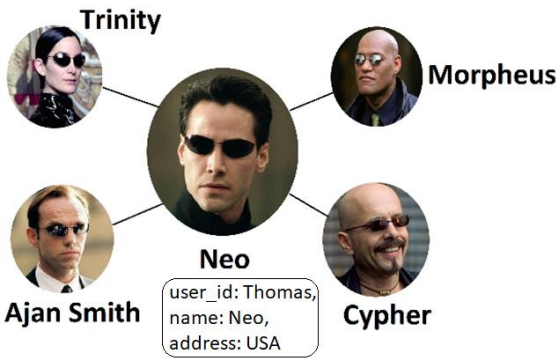

Figure 2 *The above graph shows the relationship between people that* are shown *close are they in terms of Facebook friendship.*

An undirected graph is a graph, that is, it is a set of objects sometimes called nodes that are connected, where all the edges are bidirectional. That is to say, an undirected graph contains an unordered pair of vertices.
An adjacency matrix is just a square matrix that is used to represent a finite graph. The elements of the matrix indicate whether particular pairs of vertices are adjacent to each other or not concerning the graph. Within the extraordinary case of a limited straightforward graph, the adjacency matrix is a (0,1) matrix with zeros on its diagonal.
Similar to an adjacency matrix, an adjacency list is a collection of unordered lists that are being used to represent a finite graph. Each list defines the set of neighbors of a vertex within the graph.
The best reason behind the utilization of the adjacency list within the above case is that it is exceptionally simple to create the graphs utilizing the unordered list with the help of the adjacency list. It will also become easy to relate between so many users and how they are connected. As we say about the adjacency list we are moving towards the vertices and the collections which is the reason why it is so good to use the adjacency list in the above situations. It will be best to associate the vertex of the graphs and the whole collective neighboring of the vertices of edges to be connected and give out the relation in the form of a graph. Hence, this is the reason why we must use an adjacency list for the above task and it will be very easy and useful to implement such things. In the case of Facebook, the user base is in the order of billions. Considering an adjacency matrix for storing the connections for all the users. Let's say we have 1 billion users. We will need a (N x N) matrix to store all the edges. The space complexity here will

be O(N2), i.e. (1 billion) 2 which is too large to store practically. If we use an adjacency list, then the number of edges that we have to store will only be limited to the number of connections we have between users. If A is friends with B and C then we will only store the edges for B, C in A's list and A, in B's and C's list. We do not have to store anything for the other users and that greatly reduces the space required to store the edges. A walk is a sequence of vertices and edges of a graph that is, if we were to travel a graph then we would get a walk. Vertex and edges can be repeated. Walks can be open or closed. A trail is a path with the first vertex and last vertex as its endpoints. A trail is said to be closed on the off chance that its endpoints are the same. A path in a graph may be a limited or interminable grouping of edges that joins an arrangement of vertices which, by most definitions, are all unmistakable.
A bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets such that no two graph vertices within the same set are adjacent.
Depth-first search (DFS) is an algorithm that starts at the root vertex, often selecting any node as the root node in the case of a graph, and explores as far as possible along each branch before coming back to the root node and repeating. DFS has less step cost than BFS but DFS may not always find the optimal path.
Breadth-first search (BFS) is an algorithm that starts at the tree root and explores all of the neighboring nodes at a certain depth before moving on to the nodes at the next depth level. BFS has more step cost than DFS but BFS always finds a shorter path than DFS.
Breadth-first search is the algorithm that guarantees us in finding the correct number of degrees of separation between any two people in a social network group.
A random network consists of N nodes where each node pair is connected with the probability p. Used to study the probability distributions of various networks.

A scale-free network is a connected graph or network with the property that the number of links originating from a given node exhibits a power-law distribution.

Anatomy of The Graph Theory in Social Networks

Expect you have got a Facebook application in which everybody has an account and two individuals will interface once they end up buddies. Let’s expect you to illustrate the accounts and their associations employing a graph data structure. We described how you implement the graph in a few sentences, e.g. directed or undirected? What are vertices? What are the edges, etc?

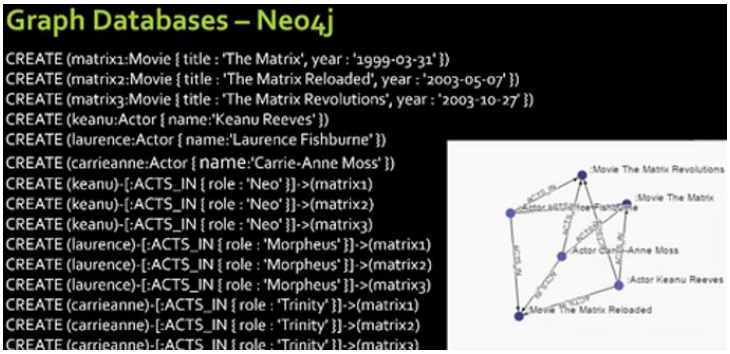

Figure 3 *The showing off the edges of Facebook networks in the database.*

Presently expect that Starbucks incorporates a page on your Facebook application and individuals can take after this page. How do you execute it on your Facebook? Presently expect A has an account on your Facebook and could be a Starbucks adherent. You need to check whether it is conceivable to begin from A and go over the graph and reach all other Starbucks devotees through As buddies, step buddies, step-step friends, and so. What algorithm do you use for that? Explain how you will implement the algorithm to this application. Note that you should efficiently apply the algorithm. You will check the complete graph but that takes a long time, so that's not productive

for our issue. We look for only Starbucks followers. Presently each individual enters a friendship level for each of their buddies. Once A sends a friendship request to B and once B acknowledges the friendship, each one enters a friendship level for this modern buddy. The friendship level of B for A is a number within the extent (1-5), and it is based on the number of times A meets, writes, or calls B. In case of C is not a coordinate buddy of A but both are buddies with B, the friendship level of C for A is the whole friendship level of B for A-plus the friendship level of C for B. In case there are numerous individuals between A and X, the friendship level of X for A is the whole of the friendship levels along the way from A to X. You need to discover the least friendship level of X for A. Type in down an algorithm that finds the esteem and the way for the least friendship level of X for A. What type of graph do you use for this algorithm: directed or undirected? What is the time complexity of this algorism? Define. Does the algorism allow you to reply to the address otherwise you still have to translate the yield of your algorism to induce the way and least friendship level of X for A? In arrange to appear the individual account on Facebook as a vertex and an edge between two people on the off chance that they are buddies, at that point, an undirected graph is way better data structure because friendship in Facebook is a symmetric relation. Similarly to show the Starbucks page in Facebook, this page can be one of the vertexes of the graph and there is an edge between vertex corresponding to the Starbucks page to vertex corresponding to a person's Facebook account if the person follows the Starbucks page. We can mark a label "S" on the vertices corresponding to those Facebook accounts, who are followers of Starbucks. If we want to find the people only by going from person A account, then the best algorithm to do this task is to perform Breadth-First Search on this graph starting from vertex corresponding to the Facebook account of A. So we perform BFS starting from A and in each step, we explore the A's friend account and then step-friends and so on and select those vertices as Starbucks followers, whose vertex label is "S" which we did in the previous question. Thus we will keep the total count of the number of vertices explored by BFS starting from A whose label was marked with "S". Now if the total count of vertices explored in BFS with label "S" is equal to the number of followers of Starbucks then the answer is "yes", it is possible to find all the Starbucks followers by exploring from A's account, and "no" otherwise. This problem of finding the path for the minimum friendship level of X for A is equivalent to finding a minimum weighted path from X to A. This problem can be represented by a weighted undirected graph since friendship level is a symmetric relation and the weight of an edge is equal to friendship level between the vertices connected by the edge. So now the problem of finding the minimum friendship level between X and A is equivalent to finding the minimum weighted path from X to A which can be done by using a single-source shortest path algorithm i.e. Dijkstra algorithm by selecting either of vertex X or A as the source vertex and performing the Dijkstra algorithm to get the shortest distance between X and A. The value of the shortest distance from X to A will be the minimum value of friendship level between X and A and the path in the tree formed after the Dijkstra algorithm will be the minimum weighted path. Thus we will get the solution. The time complexity will be equal to that of the Dijkstra algorithm, which is O (E log V). Note that since the friendship level is between a range [1-5], but the path returned by the Dijkstra

algorithm may exceed the value, but our concern is minimizing the sum of friendship level in the path from X to A, so we will not consider the outcome of friendship level from X to A that it is exceeding the range [1-5] or not.

Applications of Graph Theory in Social Networks

The thought of social networks may be an important concept that's pertinent to social science research, graph theory, and measurements. On an abstract level, it refers collectively to the ties among individuals or groups in the society (known as social actors). It can be analyzed by studying the social interactions of a particular social actor with other individuals or organizations in society. The concept can be understood better by visualizing oneself as being embedded in a circle of other people or groups to whom one is connected by branches. A suitable operational definition of "social networks" requires a detailed exposition of how such networks can be "measured" or analyzed. Its nature and properties should be described so that the concept becomes accessible to a larger group of people.

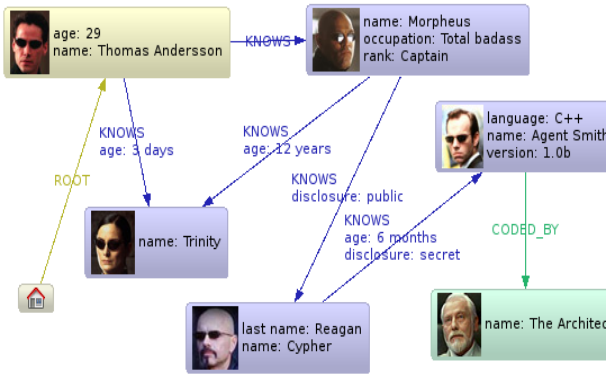

Figure 4 *The connection between a large group of people.*

Operationally, social networks include ties such as friendships between individuals, workplace relationships, family relations, sexual relationships, kinship, relations between organizations, and so on. One method of determining social networks is by choosing a group of social actors whom we know in advance. Each social actor is then shown the list of other actors and asked to name friends, relatives, advisers, opponents, subordinates, and superiors with whom he or she interacts. In this manner, the ties between the given set of social actors can be determined.

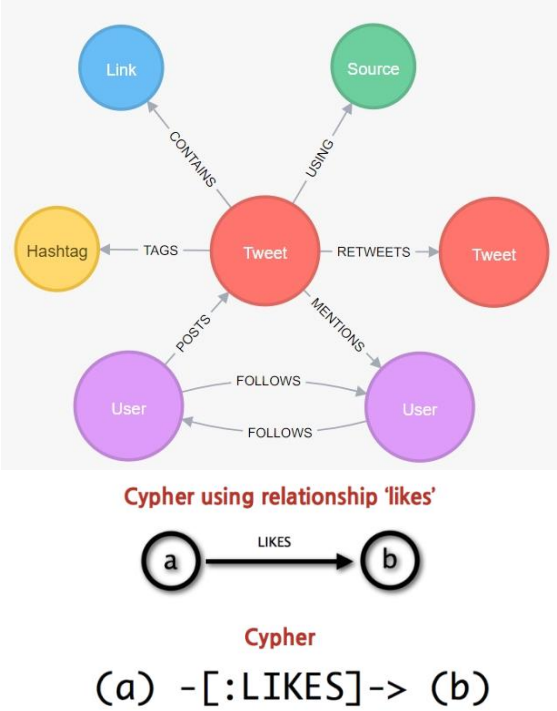

Figure 5 *The Graph Model we use to load your data such as on Twitter.*

Social networks begin at the micro-level with the individual and involve tracing of simple relationships such as family and romantic relationships to more complicated relations at the social level such as cultural kinship. Analysis at the meso-level involves tracing formal and informal relationships within organizations. Finally, at the macro-level, the implications of the interactions on a larger population are measured, instead of the interpersonal interactions themselves. Social networks can be pictorially represented through the use of sociograms in which people are represented as points or nodes and ties are represented as lines connecting the points. In almost all the social networking sites, graphs are used. In Facebook, users are considered to be the vertices. Two or more users are connected on Facebook if they are friends. Thus, an edge is used to

represent the relation between two friends. In general, if two or more people are connected on a social networking site then an edge is present between those two vertices. Facebook is an example of an undirected graph.

Figure 6 *Example of an undirected graph that more users are connected on Facebook.*

In programming, graphs are used to represent the flow of computation. For example, Google maps use graphs for building transportation systems, where the intersection of two (or more) roads are considered to be a vertex and the road connecting two vertices is considered to be an edge, thus their navigation system is based on the algorithm to calculate the shortest path between two vertices. In Facebook, users are considered to be the vertices and if they are friends then there is an edge running between them. Facebook's buddy recommendation algorithm uses graph theory.
In databases, graphs are used to manage, visualize, and analyze the huge information and complex relationships. They provide a simplified description of scenarios data sets in a way that produces a useful understanding of complicated data. Graphs empower clients to demonstrate data precisely as they are spoken to within the real-world situation, this will altogether upgrade the operations on data. In this way, graphs can keep all the data approximately and protest in a single node and show the related data by connections associated with it. Queries can be developed based on the graph structure. For occurrence, the finding of the shortest way can be considered as a subgraph from the initial graph. Operationally, graphs can be put away proficiently inside databases utilizing extraordinary graph capacity structures, and utilitarian graph algorithms for the application of particular operations.

Graph Databases and Neo4j

Neo4j refers to the graph database management system. It is a NoSQL database such as MongoDB, Neo4J, and Berkeley DB. It is an open-source platform and is thus available for free. It provides the ACID transactional properties for your applications. Neo4j allows concurrent connections.

We use Neo4j because of the following advantages:

- It is used for fraud detection for the Master Data Management Systems.
- It was used for the recommendation engine and product recommendation system.
- Real-time insights
- High availability
- Flexible data model
- Easy retrieval
- No joins
- Connected and semi-structured data
- Cypher query language

The notable features of Neo4j:

- ACID (Atomicity, Consistency, Isolation, Durability) properties
- Scalability
- Flexibility
- Data model
- Built-in web application

- Cypher query language

On the other hand, Neo4j has some disadvantages:

- A relational database is more secure as compared to the Neo4J graph database. Relation databases are also more mature rather than graph databases.
- The Neo4J graph database does not provide any kind of built-in mechanism of function for managing the security. The Neo4J graph database is more flexible but not more secure as compared to relational databases.

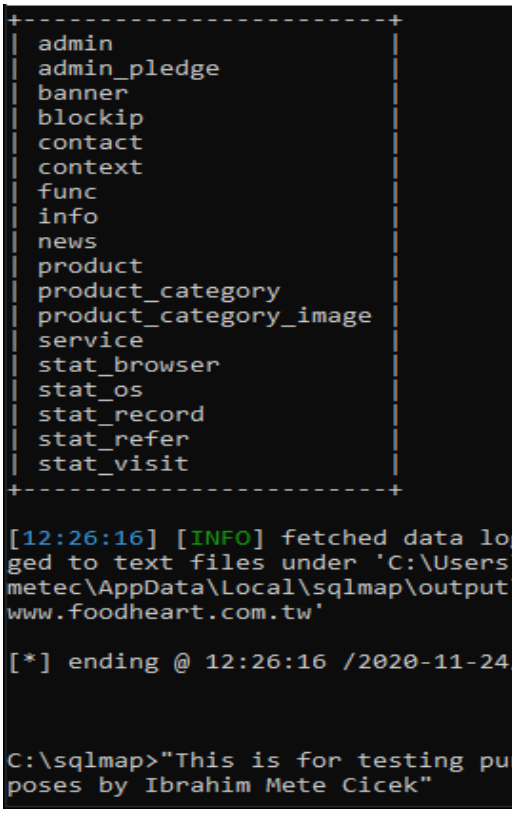

Figure 7 *Sqlmap tool that displays admin bypass in the database using SQL injection in Python driver.*

SQL injection is a common hacking method used to attack databases. An attacker can steal and change the information of users registered in the database with this method.

Neo4J focuses more on the relationships between values than the commonalities among sets of values because we know that the relationships between the nodes are the key part of a graph database. And it allows for the finding of the related data.

Number	Source	Target
0	Morpheus	Trinity
1	Morpheus	Neo
2	Morpheus	Cypher
3	Neo	Trinity
4	Neo	Morpheus
5	Neo	Cypher
6	Neo	Ajan Smith
7	Cypher	Morpheus
8	Cypher	Neo
9	Ajan Smith	Neo

Table 1 *The Graph Database Structure that displays a user's links on Neo4j.*

The relationships connect the two nodes. Also, the cypher plays an important role in building and analyzing the graph. The Neo4J's cypher is the graph query language the allows the users to store and retrieve the data from the graph database.

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 22 23:17:39 2020

@author: İbrahim Mete Çiçek
"""
from neo4j import GraphDatabase

class Friend:
    def __init__(self,name,location):
        self.name = name
        self.location = location
        self.limited = 5
        self.friends = []
    def __str__(self):
        return "%s - %s"%(self.location,self.name)
    def friend_add(self,ogr):
        if self.limited > 0:
            self.friends.append(ogr)
            self.limited -= 1
        else:
            print("Full limited")
    def friend_list(self):
        print("%s Added friends:"%self.name)
        for ogr in self.friends:
            print(ogr)
class friend:
    def __init__(self,name,surname,number,age):
        self.name = name
        self.surname = surname
        self.number = number
        self.groups = []
        self.age = age
    def __str__(self):
        return "%d , %s, %s"%(self.number,self.name,self.surname)
```

Figure 8 *The Programming that displays find user's nodes using Cypher in Python driver.*

Neo4j data is the record and the data in a graph database. Nodes are used to represent entities. We use the CREATE statement to create a Node in the Neo4J. We use Neo4j's MATCH because the MATCH clause allows us to specify the patterns, it searches for the database. And with the help of the MATCH clause, we can get the data into the current set of bindings. We create an index using the Neo4J because as we know that the index is a data structure that improves the speed of the data retrieval operations in the database. When we create the index, Neo4J will manage it and the important thing is that it also keeps updating whenever the database is changed. We use the constraints in the Neo4J because it ensures that our database will never contain more than one property value and with a specific label and it also confirms that the data adheres to the rules of the domain. As we know that the Cypher supports the queries that are the parameters that are submitted as JSON. The Neo4j gets its standalone server mode, based on the interaction via REST. As we know that the High Availability clusters are made up of at least three Neo4j instances: it can be the one "master" and the two "slave" instances.

The master instance performs writes and it also pushes the data out to slave instances with the successful write completion. The member can be a master or a slave. both are identical hardware. A neo4J master election is an approach that provides the solution of High Availability across the cluster of machines, this is for maximum read scaling and reliable uptime. Neo4J master election is the election that occurs during the normal running of the cluster and they do not have to pose an issue in and of themselves. This is the first public release of the Neo4j-Graph-Algorithms library. Our thanks go to Martin Knobloch, Paul Horn, and Tomaz Bratanic for all the development and documentation work on this project. Numerous clients communicated intrigued in running graph algorithms specifically on Neo4j without having to utilize an auxiliary framework. We too tuned these algorithms to be as proficient as conceivable. These algorithms speak to client characterized methods which you will be able to call a portion of Cypher articulations running on the best of Neo4j.

Figure 9 *The graph database that* displays *a user's links on Neo4j.*

The Algorithms in Neo4j

The algorithms covered by the library are:

- Centrality:
  Page Rank
  Betweenness Centrality
  Closeness Centrality
- Partitioning:
  Label Propagation
  (Weakly) Connected Components
  Strongly Connected Components
  Union-Find
- Path Finding:
  Minimum Weight Spanning Tree
  All Sets and Single Source, Shortest Way
  Multi-Source Breadth-First-Search

Our group contains Client named nodes that can have a buddy relationship with other clients. Clients moreover type in audits and recommendations on businesses. All of the meta-data is stored as properties of nodes, except for categories of the businesses, which are represented by separate nodes labeled Category. Graph models continuously depend on the application we have in intellect for it. Our application is to analyze networks with graph algorithms. If we were to utilize our graph as a suggestion engine, we might build a diverse graph model.

Using Graph Algorithms in Neo4j

This subtopic provides explanations and examples for each of the algorithms in the Neo4j Graph Algorithms library. Graph algorithms are utilized to compute measurements for graphs, nodes, or connections. They can provide insights on relevant entities in the graph or inherent structures like communities. Many graph algorithms are iterative approaches that frequently traverse the graph for the computation using random walks, breadth-first or depth-first searches, or pattern matching. Due to the exponential development of conceivable ways of expanding separate, numerous of the approaches to have tall algorithmic complexity. Luckily, optimized algorithms exist that utilize certain structures of the graph, memoize as of now investigated parts, and parallelize operations. Whenever possible, we have applied these optimizations.

The 2 traversal algorithms which are used in Neo4j are explained below.

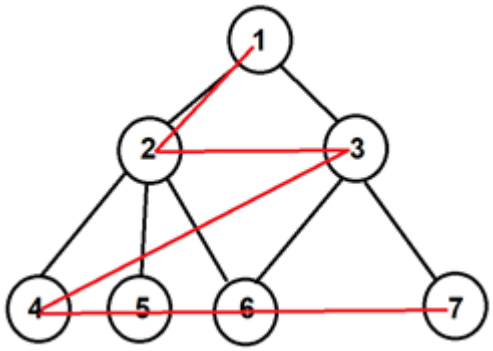

Figure 10 *Breadth First Traversal*

As the name implies, in this traversal algorithm after traversing a node, we first move across all the adjacent nodes to it, thus we move across the breadth or the next level. That is why it is called the level-order traversal. The algorithm for it uses a queue. We first pop a node of a graph from the queue and mark it visited, then we push all its adjacent nodes if they are not already visited. As in the above illustration after marking 1 as visited we will push 2 and 3 into the queue, then we will pop 2 and push its adjacent nodes i.e. 4,5,6 then we will pop 3 and push 7(6 is already visited). Then we will pop 4,5,6 and 7. Thus the breadth order traversal for the above graph starting from 1 would be 1,2,3,4,5,6,7.

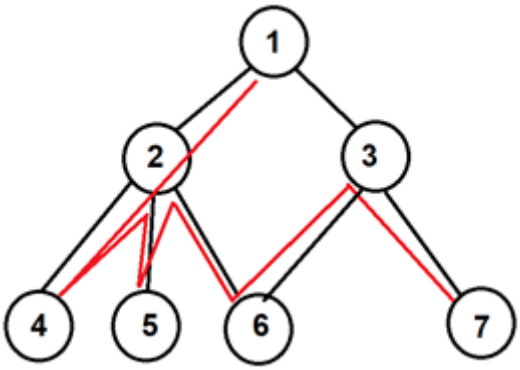

Figure 11 *Depth First Traversal*

Again as the name implies, we first traverse along the depths. In this traversal instead of completing levels, we move down the hierarchy. This is done using a stack. Since the algorithm uses a stack it also has a recursive implementation. The stack-based implementation is similar to Breadth-first traversal, but it uses a stack instead of a queue. In a recursive implementation, we first mark the current node as visited and then recur for all adjacent nodes. In the above illustration, we will first pop 1 from the stack then push 3 and 2 into the stack, then we will pop 2, and then push 6, 5, 4. Then we will pop 4,5,6 and then 3, following which 7 will be pushed and popped. Thus the depth-first traversal becomes 1,2,4,5,6,3,7.

The Using of Neo4j in social networks

We can project the graph we want to run algorithms on with either label and relationship-type projection, or cypher projection.

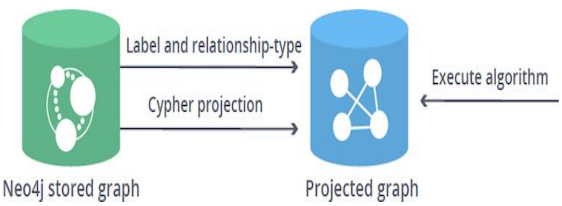

Figure 12 *displaying operations of stored graphs on Neo4j.*

The anticipated graph show is isolated from Neo4j's put away graph model to empower quick caching for the topology of the graph, containing as it were significant nodes, connections, and weights. The anticipated graph model does not back numerous connections between a single combination of nodes. During projection, only one relationship between a pair of nodes per direction (in, out) is allowed in the directed case, but two relationships are allowed for BOTH the undirected cases.

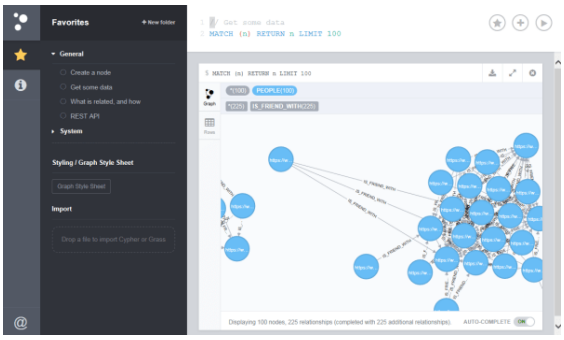

Figure 13 *The Graph model of Facebook that* displays *a user's links on Neo4j.*

Figure 14 *The Graph model of Twitter that displays a user's links on Neo4j.*

We will explore a simple application of Graphs in Neo4j. Namely, on Facebook, or other social media sites, you may have encountered automatic friend suggestions. For example, Linked In is always suggesting connections of connections that I might want to add to my network. If two people are friends, we can imagine that they are adjacent in the network. If two people are not friends, they could still be connected through a mutual friend, but they would not be adjacent. However, if we both have a mutual friend, it stands to reason that we might also befriend (or might want to at least add each other as friends) specifically, if we will write a

suggested friends method that, when to run on a given node, will determine new friends to suggest for that node. If nodes A and B are friends, and nodes B and C are friends, but nodes A and C are not friends, and we run the suggested new friends method on A, C should be a suggestion for A. A social network can be spoken to by a graph where vertices speak to people and edges speak to a buddy relationship between two people. A list of 5 people and their buddies are given underneath.

Friends of Trinity: Neo, Morpheus
Friends of Ajan Smith: Neo
Friends of Cypher: Neo, Morpheus
Friends of Neo: Trinity, Ajan Smith, Cypher, Morpheus
Friends of Morpheus: Trinity, Cypher, Neo

A social network is represented by a graph that depicts the friendship between 5 persons, namely, Trinity, Ajan Smith, Cypher, Neo, and Morpheus. persons, namely, Trinity, Ajan Smith, Cypher, Neo, and Morpheus.

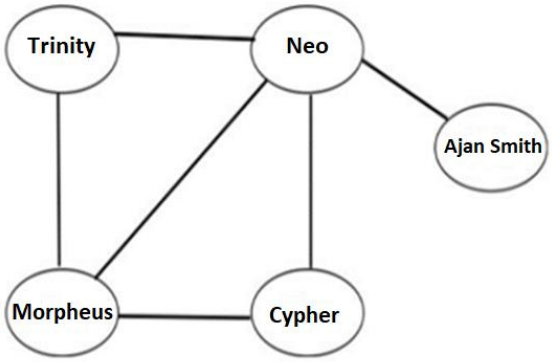

Figure 15 *In the graph, the nodes represent persons and edges represent a friendship between two persons as Facebook and LinkedIn.*

	Trinity	Ajan Smith	Cypher	Neo	Morpheus
Trinity	-	-	-	1	1
Ajan Smith	-	-	-	1	-
Cypher	-	-	-	1	1
Neo	1	1	1	-	1
Morpheus	1	-	1	1	-

Table 2 *The adjacency matrix representation of the above graph is as follows.*

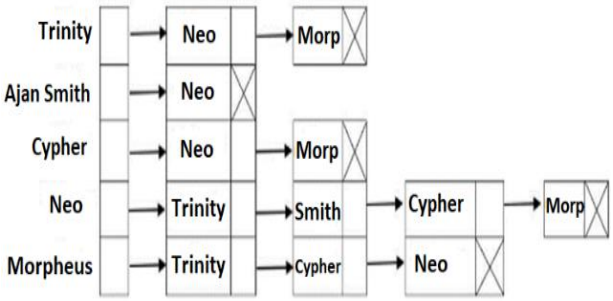

Figure 16 *The adjacency list representation of the above graph is as follows.*

Using database drivers in Neo4j

The database drivers are available to provide access to Neo4j for applications. This driver suitable for some object-oriented programming languages:

- C# works with any .NET language
- Java works with any JVM language
- JavaScript
- Python

The driver API is planning to be topologically skeptical. By this, we mean that the underlying database topology, single instance, causal cluster, etc. can be altered without requiring a corresponding alteration to the application code. Within the common case, only the association URI ought to be adjusted when changes are made to the topology.

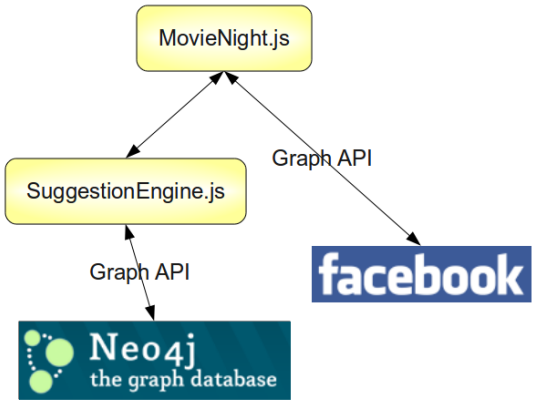

Figure 17 *The showing off the working schema of database drivers in Neo4j.*

The drivers do not support HTTP communication. If you need an HTTP driver, there are several community drivers to choose from. See also Chapter 5, HTTP API.

Coronavirus Tracker

We can use the Neo4j database system to do a coronavirus tracker map that detects sick people with a pandemic. In this way, we can prevent the pandemic and protect people's health by learning the regions where the epidemic is intense and taking measures accordingly.

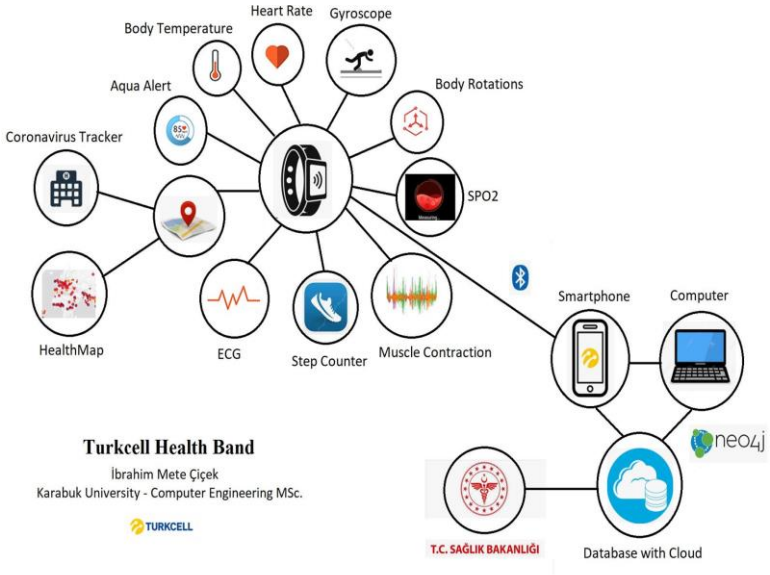

Figure 18 *Turkcell Health Band with coronavirus tracker.*

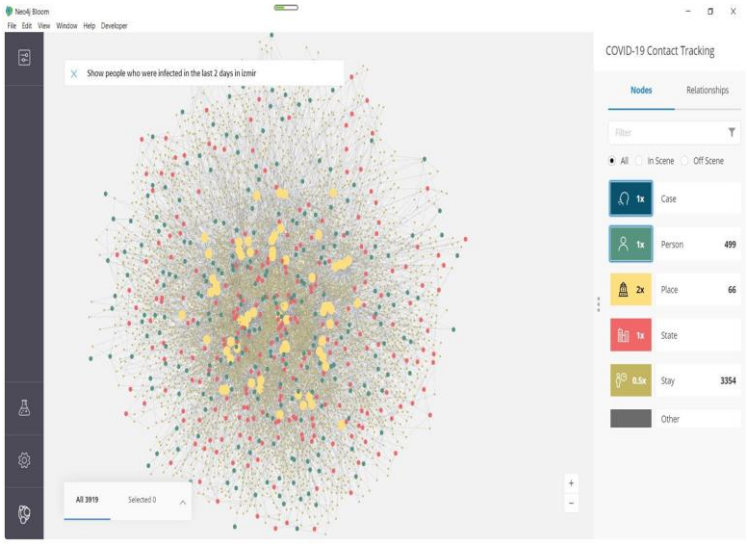

Figure 19 *It shows the disease intensity in the area where the selected person is on the map in the picture.*

In this way, we can prevent the pandemic and protect people's health by learning the regions where the epidemic is intense and taking measures accordingly.

Conclusions

To sum up, this has been a brief introduction to graphs on social networks. We presented several of the most popular models of Graph Theory in Social Networks. We have examined these models to create databases in social networks. We learned that Neo4j is the most popular graph database system in the world because Neo4j is used in social networks such as Facebook, Twitter, and LinkedIn. After all, it is the fastest database compared to other types. We learned Neo4j is a graph database management system. It uses property graphs to extract the added value of data. The performance is agile, flexible, and scalable. It is an open-source, NoSQL, native Database. It provides ACID. The data is stored exactly on the whiteboard. The database pointers to navigate and traverse the graph. Neo4j is written in Java language. The Cypher that is Neo4j's programming language can be coded in the drivers of other object-oriented programming languages. It has many advantages but it has some security bugs as compared to Relational databases such as SQL injection. In this review paper, you have learned what a graph is and how to utilize a graph to imagine the social network you can analyze data about other social networks you care about such as LinkedIn, Twitter, and Facebook to analyze the network between you and your friends or followers. You can get to practice using the graph as well as understand your relationship with your friends. You will learn information such as whether you know them directly or through their connections with the others in social networks using the graph theorem. Also, we have seen how we can actively providing advantages in many different areas and important projects in the sector by using the Neo4j database system.

References

1. Sasaki, B.Y., Chao, J., Howard, R. H. *Graph Databases for Beginners.* Retrieved from https://go.neo4j.com/rs/710-RRC-335/images/Graph_Databases_for_Beginners.pdf

2. Easley, D., Kleinberg, J. (2010). *Networks, Crowds, and Markets*. Cambridge University Press. Retrieved from http://www.cs.cornell.edu/home/kleinber/networks-book/

3. Sun, C. (2018). *A Natural Language Interface for Querying Graph Databases.* Massachusetts Institute of Technology. Retrieved from https://dspace.mit.edu/bitstream/handle/1721.1/119708/1078222310-MIT.pdf

4. Mpinda, S. A. T., Maschietto, L. G. (2015). *Graph Database Application using Neo4j*. International Journal of Engineering Research & Technology. Retrieved from https://www.ijert.org/research/graph-database-application-using-neo4j-railroad-planner-simulation-IJERTV4IS041188.pdf

5. Dr. Selmer, P. (2018). *Engineer at Neo4j and member of the open Cypher Language Group. NoSQL, graph databases & Cypher.* Retrieved from http://www.dcs.bbk.ac.uk/~ap/teaching/ADM2018/PetraSelmerADMLecture2018.pdf

6. Chen, C., Richards, B., Wang, M., Zhong, A. *Graph Databases.* Retrieved from https://cs.brown.edu/courses/cs227/archives/2013/slides/graph.pdf
7. Needham, M., Hodler, A.E. (2019). *Graph Algorithms Practical Examples in Neo4j*. Retrieved from https://go.neo4j.com/rs/710-RRC-335/images/Neo4j_Graph_Algorithms.pdf

8. Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press. Retrieved from http://math.sjtu.edu.cn/faculty/xiaodong/course/Networks%20An%20introduction.pdf

9. Nguyen, H. (2017). *Networks: CIS 700/002: Special Topics: sqlmap-automatic SQL injection and database takeover.* Department of Computer and Information Science. The University of Pennsylvania. Retrieved from https://rtg.cis.upenn.edu/cis700-002/talks/sqlmap.pdf

10. Anley, C. (2002). *Advanced SQL Injection In SQL Server Applications.* An NGSSoftware Insight Security Research (NISR) Publication. Retrieved from https://crypto.stanford.edu/cs155old/cs155-spring11/papers/sql_injection.pdf

11. Elmasri, R., Navathe, S., B. (2016). *Fundamentals of Database Systems.* Retrieved from https://www.cs.purdue.edu/homes/bb/cs448_Fall2017/lpdf/Chapter24.pdf

12. Gessert, F., Wingerath, W., Friedrich, S., Ritter, N. *NoSQL Database Systems: A Survey and Decision Guidance.* Universität Hamburg, Germany. Retrieved from https://www.cs.purdue.edu/homes/bb/cs448_Fall2017/lpdf/Chapter24.pdf

13. Silvescu, A., Caragea, D., Atramentov, A. *Graph Databases.* Department of Computer Science. Iowa State University. Retrieved from http://people.cs.ksu.edu/~dcaragea/papers/report.pdf

14. Chen, Z., Pu, Y., Shelden, D.R. *A Graph Database and Query Approach to IFC Data Management.* School of Computational Science and Engineering, Georgia Institute of Technology.

15. WILSON, C., SALA, A., PUTTASWAMY, K. P. N., ZHAO, B. Y. *Beyond Social Graphs: User Interactions in Online Social Networks and their Implications.* University of California Santa Barbara. Retrieved from https://sites.cs.ucsb.edu/~ravenben/publications/pdf/graphs-tweb12.pdf

16. Narayanan, A., Shmatikov, V. *De-anonymizing Social Networks.* The University of Texas at Austin. Retrieved from https://www.cs.utexas.edu/~shmat/shmat_oak09.pdf

17. Campbell, W. M., Dagli, C. K., Weinstein, C.J. *Social Network Analysis with Content and Graphs.* Massachusetts Institute of Technology. Retrieved from https://www.ll.mit.edu/sites/default/files/page/doc/2018-05/20_1_5_Campbell.pdf

18. Li, C., Goldwasser, D. *Encoding Social Information with Graph Convolutional Networks for Political Perspective Detection in News Media*. Purdue University. Retrieved from https://www.cs.purdue.edu/homes/dgoldwas//downloads/papers/LiG_acl_2019.pdf

19. He, H., Singh, A.K. *Graphs-at-a-time: Query Language and Access Methods for Graph Databases.* University of California, Santa Barbara. Retrieved from https://sites.fas.harvard.edu/~cs265/papers/he-2008.pdf

20. Fan, W. *Graph Pattern Matching Revised for Social Network Analysis.* University of Edinburgh & Harbin Institute of Technology. Retrieved from http://homepages.inf.ed.ac.uk/wenfei/papers/icdt12.pdf