DOI 10.20544/TTS2021.1.1.21.p08
UDC••656.11:711.4]:[004.652:519.179
Original scientific paper

Graph Database Modeling of Urban Area Road Networks

Ilija Hristoski[1] and Marija Malenkovska Todorova[2]

Abstract – **Specific features delivered by the class of Graph Databases allow for modeling road networks of different scales. In this paper, we propose and implement a Graph Database model of a typical road network inherent to urban areas. Such a database can be used along with diverse information systems, to provide information of various natures.**

Keywords – **Road networks, Urban areas, Traffic, Graph databases, Neo4j.**

I. INTRODUCTION

Back in 1736, Leonhard Euler, the great Swiss mathematician, successfully resolved what later appears to become a notable historical problem in mathematics, the famous "The Seven Bridges of Königsberg" problem, and so laid the foundations of topology and graph theory [1]. Almost three centuries later, graphs have been rediscovered with the emergence of graph databases, which make a rapidly growing segment of modern non-relational databases.

The rapid progress of urbanization has induced, besides all known benefits, a lot of complex, yet highly important issues to be concerned about, such as traffic congestion and pollution. Inner human migrations that happen on a daily basis have pointed out the necessity of efficient and reliable transportation, as well as enlarged and enriched urban road network infrastructure. Effective traffic management that would lead to improving traffic flows and reducing both the congestion and pollution in urban areas needs a large amount of diverse data to be acquired and stored properly and processed quickly.

Inspired by Euler's approach to representing geographical topology with graphs, and having minded the need of effective traffic management, the paper focuses on exploring the possibilities of representing urban area road networks with state-of-the-art graph databases. More precisely, the aim is to illustrate the procedure of manual building of such a database from scratch, as well as to point out the vast potentials and benefits of their application in a modern traffic science.

The rest of the paper is organized as follows. The most recent research related to the substance elaborated hereby is given in Section II. Section III provides a brief overview of graph databases, their basic building blocks, and their most notable characteristics. The analysis of data requirements is given in Section IV. Section V focuses on the implementation specifics using Neo4j and CQL. The last section concludes.

[1]Ilija Hristoski is with the "St. Kliment Ohridski" University – Bitola / Faculty of Economics, Prilepski Braniteli St 133, 7500 Prilep, North Macedonia, e-Mail: ilija.hristoski@uklo.edu.mk

[2]Marija Malenkovska Todorova is with the "St. Kliment Ohridski" University – Bitola / Faculty of Technical Sciences, Makedonska Falanga St 37, 7000 Bitola, Republic of North Macedonia, e-Mail: marija.malenkovska@uklo.edu.mk

II. RELATED RESEARCH

The application of graph databases in technical sciences, especially in the sphere of transportation and traffic sciences, has become mainstream during recent years. Some of the most notable research endeavors are being attributed to Czerepicki, who sheds light on the perspectives of using NoSQL databases in intelligent transportation systems [2], and also presents an innovative concept of applying graph databases in transport information systems, by implementing the solution of finding the optimal route between two stops in a public transport environment [3]. In addition, Zheng *et al*. proposed a spatio-temporal data model based on a graph database, which integrates the three temporal GIS's key elements: space, time and attributes [4]. The urban data integration (UDI) framework, which is capable of integrating heterogeneous urban data stored in a graph database, has been introduced in [5]. The concepts, methodologies, and applications of urban computing, including many transportation issues, have been elaborated in [6]. The usage of the Neo4j graph database in modeling urban traffic has been explained in [7].

III. GRAPH DATABASES

Graph databases belong to the family of NoSQL databases. Thanks to the fact that "in graph databases, relationships are considered to be 'first-class citizens'", they address one of the great macroscopic business trends of today: "leveraging complex and dynamic relationships in highly connected data to generate insight and competitive advantage" [8]. They organize and store data in the form of a graph, based on the mathematical principles of the graph theory. Fundamentally, a graph can be considered as a collection of nodes (vertices), graphically depicted by circles/bubbles, and edges (arcs), portrayed by directed arcs connecting two nodes. Nodes typically represent entities (i.e. particular instances of entity types), whilst edges are used to represent various relationships between those entities. Both nodes and edges can hold detailed data (i.e. attributes) in a 'property/value' manner, to describe the entities represented by nodes and to depict the nature of relationships among them, respectively.

Neo4j is the pre-eminent and pre-dominant graph database engine, offering ACID transactions, native graph data storage (it is fully optimized and designed for storing and managing graphs), and graph data processing, i.e. an index-free adjacency (connected nodes physically 'point' to each other in the database). It supports the Cypher query language (CQL), which aims to use an 'ASCII-art'-like style syntax to make storing and querying of graph data as easy as possible.

Graph databases are inherently schema-less and are characterized by high efficiency, scalability, and ability to handle a large number of concurrent users.

IV. ANALYSIS OF DATA REQUIREMENTS

Instead of transforming the relational database schema or the E-R diagram of a real (existing) or generic (non-existing) transport information system into a graph database model through the process of logical mapping, the process of physical mapping of urban area road network directly into a corresponding graph database model is being exploited and elaborated in this paper instead. It can be carried out in two, mutually exclusive, and inverse, ways, a primal way and a dual one, which are both based on the usage of urban area road maps. From a graph database perspective, the basic features of these two approaches are the following ones:

- The primal way; Road intersections are modeled as nodes (graph vertices), and particular road sections are modeled as relationships (graph arcs, graph edges) between two consecutive nodes;
- The dual way; Particular road sections are modeled as nodes, and road intersections are modeled as relationships connecting two consecutive nodes.

In this paper, the focus is put on the primal way, taking into account the following assumptions:

- An urban road network consists of roads and road intersections;
- A road consists of one or more road sections;
- A road section is a segment (i.e. fragment) of a road connecting two consecutive (i.e. adjacent) road nodes on a road map; The traffic on a road section can occur in one or two directions; Each direction can include one or more traffic lanes; In a graph database, a road section corresponds to a relationship connecting two consecutive nodes;
- A road node is a point on a road map, which can be either a terminal node (i.e. a road breaking point at the border of a road network map, and a beginning/ending of a road inside the road network map) or a junction node (i.e. a road intersection inside the road network map); Road nodes connect road sections into a road network topology; In a graph database, a road node corresponds to a node;
- A road intersection is a road node that connects two or more road sections.

The equivalent general graph database model, depicting a two-way road section connecting two road nodes, is depicted in Fig. 1. This is a schematic view of the basic building blocks involved in the construction of a graph database: road nodes and road sections comprised of one or two traffic directions.

It should be also notified that, in this paper, the manual procedure of physical mapping of an urban area road network into a corresponding graph database is described. In fact, Cypher can be used to load urban road network data into Neo4j from a file, such as a CSV and JSON file, to facilitate and speed up the process of building the graph database. Alternatively, the Neo4j built-in ETL (Extract-Transform-Load) feature can be used to load such data from a JDBC-connected RDBMS. In this particular case, there was no urban road network data available in any electronic format and/or relational database for the urban area of interest, so the graph database has to be built manually from scratch.

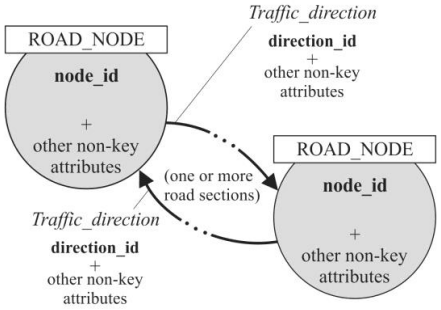

Fig. 1. Basic structure and elements of the graph database model vis-à-vis the primal way

Before implementation, one should consider what data (i.e. attributes) are going to be kept within the graph database's nodes and relationships.

A. Data Stored within Nodes

Graph database nodes should keep the following basic information: unique identifier of the node, node name, urban area zone name, city name, type of the node (e.g. terminal node - road breaking point, terminal node - beginning/ending point of a road, or a junction node - intersection point among two or more roads), type of an intersection (e.g. T-shaped, Cross-shaped, Star-shaped, Roundabout), geolocation (i.e. longitude, latitude as strings in DMS format, altitude, and geolocation data as decimal degrees.

Besides these mandatory data, graph database nodes can also include other attributes relevant to the system the graph database is intended to be used with. For instance, in the case of a pollution monitoring system based on the utilization of a distributed system of measurement stations (sensors), graph database nodes can also include attributes corresponding to data about air quality (PM_{10}, $PM_{2.5}$, NO_x, CO, CO_2, SO_2, etc.), air pressure, air humidity, air temperature, UV radiation, sonic pollution (noise), gamma radiation, etc., which can be all acquired and updated in real-time. Alternatively, all of these measurement parameters can be included in the graph database as newly added graph nodes, connected to the existing ones.

B. Data Stored within Relationships

The most relevant information that should be kept in relationships (i.e. traffic directions of road sections) is unique identifier of the traffic direction, unique identifier of the road section, unique identifier of the road (street), name of the road (street), designation of the road (street), road category, road significance, road section length, number of traffic lanes, traffic lane width, maximum speed allowed. Relationships should also include traffic flow variables, such as average flow rate, average headway, time mean speed, space mean speed, traffic density, and distribution of vehicles' relative frequencies [%] according to their category (bicycles, motorbikes, passenger cars, buses, and trucks).

V. IMPLEMENTATION

The starting point in the process of implementation is to obtain a road map of the urban area of interest, which can be generated based on a satellite image or simply manually sketched. In this particular case, Google Maps has been utilized for this task, however, other sources, like Google Earth, can be used, as well.

Further on, all road breaks at the borders of the map are being marked up with a symbol of a transparent circle, and hereby denoted clockwise with strings as '*x*1', '*x*2', …, '*xN*', $N = 13$ (Fig. 2). Additionally, all road beginnings/endings, as well as road intersections within the map are denoted with integers (e.g. 1, 2, 3, …, *M*; $M = 20$), hereby starting from left to right, and from top to bottom. Specifically, the road beginnings/endings (e.g. 2, 10, and 12) are being marked up with a symbol of a yellow-colored, double-lined circle, whilst all road intersections are being marked up with red-colored circles. Different ways of notation and markings are used in order to distinguish between the three types of circles. In Fig. 2, if circles are treated as nodes, and roads between some nodes as edges, then the resulting structure represents an undirected finite graph, *G*.

In order to achieve maximum accuracy and reliability, the act of building a graph database manually out of such an undirected graph should be carried out in a consistent and systematic way, especially in the case of complex graphs, e.g. disconnected graphs and/or graphs that include a vast number of nodes (road intersections) and edges (roads). In this particular case, we have used a slightly modified version of the Depth First Search (DFS) graph traversal algorithm that assures consistency in building up the graph database's nodes and relationships simultaneously with visiting all the edges in an undirected graph (a road map). The pseudo-code of the recursive version of the modified DFS algorithm is as follows:

```
// G: an undirected graph,
//    an input set of edges and nodes (a road map)
// u, v, w: nodes in G
// R: a directed graph,
//    an output set of edges and nodes
//    (a graph database)
procedure init_DFS(G) {
   for each u in G
      u.visited = FALSE; // on a road map
}
procedure DFS(G, v) {
   create node v in R; // in a graph database
   v.visited = TRUE; // on a road map
   for all neighbouring nodes w of node v in G
      if w.visited = FALSE then
         DFS(G, w);
      else
      {
         if edge(v, w) exists in G then
            create relationship (v, w) in R;
         if edge(w, v) exists in G then
            create relationship (w, v) in R;
      }
}
procedure main(G) {
   init_DFS(G);

   for each u in G
      DFS (G, u);
}
```

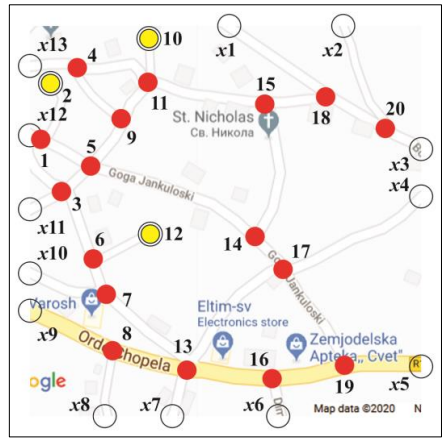

Fig. 2. Road map of the urban area of interest, presented as an undirected graph encompassing border nodes (breaking points), inner nodes (intersections and terminal nodes), and edges (roads)

The execution of the modified DFS algorithm assures visiting all nodes in graph *G*, depicted by Fig. 2. The manual generation of the corresponding graph database should be performed in parallel with the execution of the algorithm. Each labeling of a node as 'visited' on the road map means creation of a node in the graph database *R*, which can be either a terminal node resembling a road breaking point (at map borders), a terminal node resembling a beginning/end of a road (within the map), or a communication node resembling an intersection among two or more roads (within the map). On the other hand, each adding of an edge (*v*, *w*) to the resulting set of output edges and nodes, *R*, means the creation of a relationship between the nodes *v* and *w*, i.e. a traffic direction between two nodes *v* and *w* comprising a particular road segment. This way, the undirected graph in Fig. 2 becomes a directed graph, and also a visualization of the equivalent graph database layout (Fig. 3).

The corresponding graph database (Fig. 4) is comprised of 33 nodes and 74 relationships.

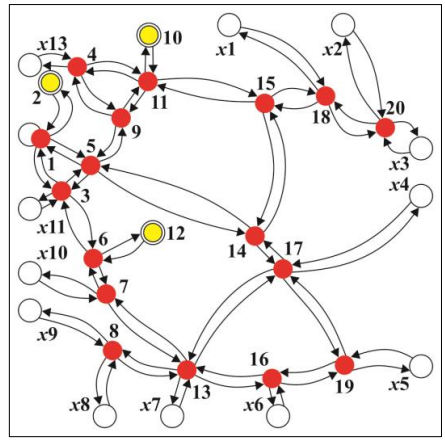

Fig. 3. Road map of the urban area, presented as a directed graph

It has been fully implemented in Neo4j using CQL (Cypher Query language).

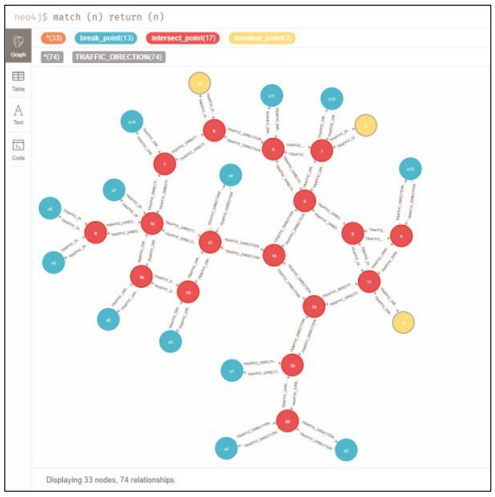

Fig. 4. Visualization of the Neo4j graph database corresponding to the road map of the urban area of interest

In Fig. 4, the structure of the graph database is logically equivalent to the directed graph of the urban area of interest, represented in Fig. 3. Moreover, the intrinsic data stored within the graph database's nodes and relationships match the facts about the existing urban road network infrastructure.

The appliance of various graph algorithms can be used to compute numerous metrics for graphs, nodes, or relationships [9-10]. For instance, given that each traffic direction within the underlying graph stores data for the attribute *RoadSectionLength* (in meters), and given that all nodes belong to the class *road_point*, the following CQL code, which embodies the *allShortestPaths* graph algorithm, computes the top 2 shortest paths between any two nodes (source, target) in a graph, in a descending order (Table I):

```
CALL
algo.allShortestPaths.stream("RoadSectionLength",
{nodeQuery:"road_point", defaultValue:1.0})
YIELD sourceNodeId, targetNodeId, distance
WITH sourceNodeId, targetNodeId, distance
WHERE algo.isFinite(distance) = true
MATCH (source:road_point)
   WHERE id(source) = sourceNodeId
MATCH (target:road_point)
   WHERE id(target) = targetNodeId
WITH source, target, distance
   WHERE source <> target
RETURN source.NodeName AS source,
   target.NodeName AS target, distance
ORDER BY distance DESC
LIMIT 2
```

TABLE I
TOP 2 SHORTEST PATHS

source	target	distance
"20"	"12"	453.0
"12"	"20"	453.0

VI. CONCLUSION

In this paper, an urban area road network has been modeled and implemented using a graph database, Neo4j, and Cypher Query Language. The powerful and expressive semantics graph databases exhibit in representing relationships among entities makes them one of the best ways to store and manage both present and future data related to traffic and transportation in urban areas.

Based on the graph database data, running CQL code can provide answers to various users' queries, such as:

- Selection of nodes and traffic directions belonging to a particular traffic path between any two arbitrary points, based on a given criterion;
- Finding out the total length of a given traffic path between any two arbitrary points;
- Finding out all the shortest paths (in number of steps or length) between any two arbitrary points in the graph;
- Implementation of various graph-based algorithms related to spatial navigation; etc.

As a bottom line, it can be concluded that, due to the unprecedented method of storing data and processing, graph databases offer a highly appropriate way to represent information for specific use in traffic and transport sciences.

REFERENCES

[1] R. Shields, "Cultural Topology: The Seven Bridges of Königsburg, 1736", *Theory, Culture & Society*, vol. 29, issue 4/5, pp. 43–57, 2012.

[2] A. Czerepicki, "Perspectives of using NoSQL databases in intelligent transportation systems", Transport, vol. 92, pp. 29–38, 2013.

[3] A. Czerepicki, "Application of graph databases for transport purposes", Bulletin of the Polish Academy of Sciences, Technical Sciences, vol. 64, no. 3, pp. 457–466, 2016.

[4] L. Zheng, L. Zhou, X. Zhao, L. Liao and W. Liu, "The Spatio-temporal Data Modeling and Application Based on Graph Database", *Proceedings of the IEEE 4th International Conference on Information Science and Control Engineering* (ICISCE 2017), pp. 741–746, Changsha, China, 2017.

[5] K. Gupta, Z. Yang and R. K. Jain, "Urban Data Integration Using Proximity Relationship Learning for Design, Management, and Operations of Sustainable Urban Systems", *Journal of Computing in Civil Engineering*, vol. 33, issue 2, 2019.

[6] Y. Zheng, L. Capra, O. Wolfson and H. Yang, "Urban Computing: Concepts, Methodologies, and Applications", *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, article 38, pp. 38:1–38:55, 2014.

[7] J. Wood, "Modeling Urban Traffic using the Neo4j Graph Database", GGE 4700 Technical Report, University of New Brunswick, Fredericton, Canada, 2015.
URL: http://www2.unb.ca/gge/News/2015/STC/Wood.pdf

[8] I. Robinson, J. Webber and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, Second Edition, O'Reilly Sebastopol, CA, USA, 2015.

[9] Neo4j, "The Neo4j Graph Algorithms User Guide v3.5", 2019, Neo4j, Inc., URL: https://neo4j.com/docs/graph-algorithms/current/

[10] M. Needham and A. E. Hodler, *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*, O'Reilly, 2019.