Graph Database Approach for Data Storing, Presentation and Manipulation

I.Fosić [*], K.Šolić [**]
[*] HEP Telekomunikacije d.o.o., Osijek, Croatia
[**] J.J. Strossmayer University of Osijek, Faculty of Medicine, Osijek, Croatia
igor.fosic@hep.hr, kresimir@mefos.hr

***Abstract* - An increasing number of IoT systems and a generation of unstructured data make it difficult to choose and apply a suitable database model. Using a NoSQL database on a test data model shows the advantages of storing, manipulating, and presenting the required data. A test data model was created on a relational database and on a graph database to compare the options of manipulating and presenting the same data. A comparison of these two models was made on a small sample, and the results were displayed on the same query as the default for both types of databases. Simpler query syntax and better visualization of data are some of the benefits presented by this model on the graph database.**

Keywords - NoSQL; Graph database, IoT, Neo4j, Cypher query language, MySQL

I. INTRODUCTION

When a database is mentioned today, one commonly refers to a relational database and a data processing system based on a relational data model. The growth of the Internet has brought on new technological challenges in various areas, such as IoT systems. This paper describes a different approach for storing, presenting, and retrieving data stored in the database. The two types of databases, relational and graph databases, have their advantages and disadvantages, and in some situations, some of the databases contribute more successfully to the result of querying data. The dynamics of the IoT system, the constant changes in the structure and the number of data are the motivation for exploring different approaches to storing and accessing data. One of the possible approaches in the dynamic IoT environment is to use the graph database. In comparing the two types of databases, MySQL was used as a relational database and Neo4j as a graph database.

One of the biggest challenges is the amount of data; more specifically, the system's difficulties in accepting and processing a large amount of data in as little time as possible. These challenges created new types of databases in which data is not organized according to the principles of the relational model, but according to a much simpler and freer data model. One of these systems are graph-based systems. Such databases consist of nodes and connections between them, and enable efficient query through nodes following their connections [1]. Graph-based databases belong to the so-called NoSQL databases. NoSQL data storage aims to be very fast and always available because it is represented in the object structure. There is no link between the tables as in a standard RDBMS (relational database management systems) so the requested objects can be accessed directly. If one is using a large system for which quick access to data is preferred, then storing data in a NoSQL database could be exactly what it takes. If one requires a consistent method of storing large amounts of data, without the need for a high response rate, perhaps using a traditional RDBMS database is a better solution [2]. A relational open source database was used for modelling data for the IoT system, as well as an open source graph database. The data model makes possible the presentation of the basic idea and the comparison of these two models on different databases, their visual presentation and the ability to retrieve the requested data. The paper is divided into six chapters. After the introduction, chapter II. describes the characteristics and advantages of the graph database over the relational database. Chapter III. describes the test environment for comparing a typical query with the relational database and the graph database. Chapter IV. describes the structure and data used in both databases. Chapter V. provides a comparison of some queries, results and their presentation. Chapter VI. contains the final considerations and suggested directions for future research.

II. GRAPH DATABASE

A graph database consists of a number of nodes which are interconnected and directed by connections. They are best suited for storing interconnected data, as they also allow for storing information about the links between the entities [3].

Most applications for IoT have to work with dynamic and fast-changing systems: new devices and applications regularly use the communication network, and must smoothly use said network. This requires a data model that can be developed without unnecessary reintegration of the database and the applications, and without affecting the availability of applications. Connections between devices and other entities can be changed faster than the data they describe. It is necessary to have a system that can easily and without stopping any part add and delete relations between entities, and graphs are the natural way of showing links between entities. Real-time response is an important factor in the Internet of Things. The graph database solves this request in a natural way, without requiring large hardware requests [4]. The graph database that was used to retrieve data is the Neo4j database, a graph platform which is specifically optimized for mapping, analyzing, storing, and retrieving network-related data to reveal invisible contexts and hidden relationships. By intuitively mapping data points and connections between them, it enables intelligent real-time applications that address today's toughest challenges including:

• Artificial intelligence and machine learning

• The Internet of Things (IoT)

• Real-time recommendations and personalization

• Detecting fraud

• Network and IT activities

• Identity management and access [5].

Graph databases are effective when used for discovering and giving meaning to complex interdependencies and relationships between entities. They are designed for easy modeling and data navigation, with extremely high performance, which makes them very popular for use on social networks. Now they are being increasingly embraced by companies that want to extract the maximum value from the Internet of Things [6]. The interconnection of devices and data focused on a scalable solution for processing requests from IoT devices has shown that the graph database is a viable solution for modeling such systems. There are many operations in IoT systems that run in real-time and require tools that can display interconnections very quickly; they use similar queries that would otherwise require JOIN operators in a relational database [7]. Modeling data in the form of a graph is simple and natural, and has the advantage of being simple to understand for common data users. The relational model is more prevalent, but when it comes to billions of devices connected to the Internet, the graph database has certain advantages in view of the complexity of the system. When the choice of a database must be applied to smaller amounts of data and a response very close to real-time reactions, then the graph database can be very useful in the IoT system [8]. The main challenge with rapidly growing amounts of data is its discrepancy. This problem has lead to a greater use of nonrelational databases that are very scalable and efficient, and can store large amounts of unstructured data. Although relational databases are able to use all three data types (structured, semi-structured and unstructured), more work and compromises are needed to achieve effective storage of unstructured and semi-structured data. Relational databases store structured data because they are already in an acceptable form, but the storage of semi-structured data involves increased complexity. Semi-structured data should first be converted into the appropriate data form for relational databases before storage. In the case of unstructured data, it is stored as a blob object and not directly [9].

III. Data model environment architecture

The data model is the set of data arranged in multiple tables in the RDBMS database, and the same pattern of data is used to create nodes and relationships in the graph database. The testing environment is a setup of software support and a virtual server. The testing environment relies on multiple physical servers in a cluster that uses the hypervisor on which a virtual server with the desktop operation system was created. The application and the database for the graph database were installed on the virtual server: Neo4j and the associated graph database. The application itself contains software support for data manipulation via the Cypher scripting language and is accessible via an Internet browser. The Internet browser connects to the application of the graph database via ports 7687 (Bolt), 7474 (HTTP), or 7473 (HTTPS). On the same virtual server, the RDBMS database was installed through the XAMMP application. XAMPP is a completely free, open source Apache distribution that contains MariaDB, PHP and the Perl scripting language [10]. The Phpmyadmin application, which connects via an Internet browser to the MySQL (MariaDB) database through port 3306, was used to access the database. The functional diagram of the server and the software support for the test environment is shown in Figure 1.

IV. Data model

Unlike classic RDBMS data models that use tables and table data display, the graph database uses a data model with a structure consisting of nodes, tags, node connections, and node properties which have the following characteristics:

A. *Nodes*

• main data elements

• connected to other nodes through connections

• can have one or more properties (attributes stored as a pair of keys / values)

• have one or more tags describing its role on the graph

B. *Connections*

• connect two nodes

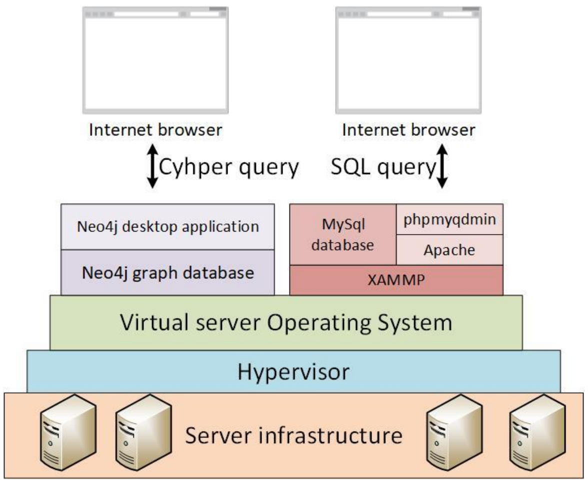

Figure 1. Test environment

• focused

• nodes can have multiple connections and can even be recursive

• can have one or more properties (attributes stored as a pair of keys / values)

C. Properties

• named values where the name (or key) is a string of characters

• can be indexed and restricted

D. Tags

• used to group nodes into sets

• a node can have multiple labels

• indexed to accelerate the finding of nodes in the graph

For the purposes of use comparison and data retrieval in the data model, an identical data model was created in the classical RDBMS database (MySQL) and in the graph database (Neo4j). The data that forms the database in the MySQL database is shown in the tables in Figure 2. The "device" table contains the device names and locations to which the device belongs (10 records), the "location" table contains the names of the locations and the region in which the location is located (13 records), and the "region" table contains the names of the regions (4 records). The "measurement" table contains the measurements obtained from the device, the date of measurement, and the type of measurement (11 records), and the "measurement_type" table contains the name of the measurement type (2 records). The data in individual tables is linked as follows: the sequence number of a record from one table is associated with the values from the second table using a foreign key. The data model in the graph database is shown in Figure 3. The data structure in the graph database followed the data structure in the relational database where five types of nodes were made: type of measurement (2 nodes, green), measurements (11 nodes, red), devices (10 nodes, yellow),

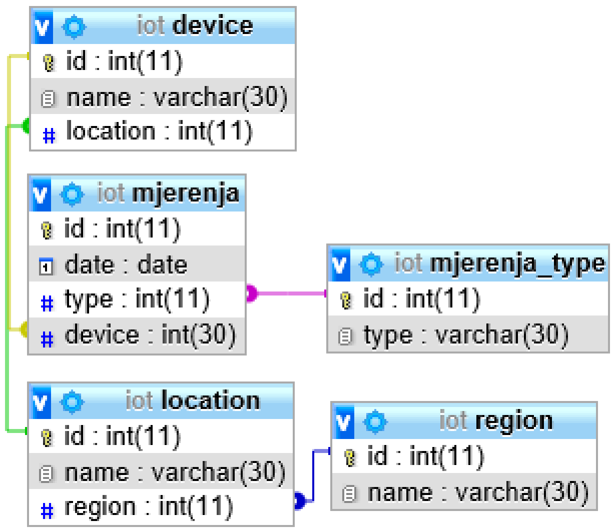

Figure 2. Model of data tables in the MySql database

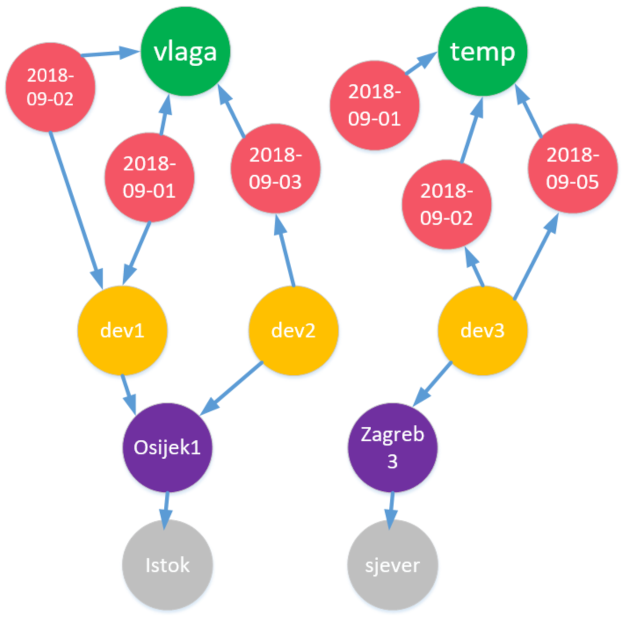

Figure 3. Segment of data model in graph database

locations (13 nodes, purple), regions (4 nodes, gray). All nodes are connected by connections, each node with a specific connection to the neighbor node. Figure 4. shows one of the relationships that is obtained by searching according to the given criteria. The image shows which types of nodes are associated with their neighboring nodes and the names of their connections (Type of measurement, Measured, Below).

Already by comparing the data display itself, it is clear that the graph displays the data more intuitively, and certain data connections can already be determined from a visual inspection. Thus, one can in a simple manner, for example, determine which devices belong to which region, which device did not deliver a measurement, during which days there had been no moisture measurement at all, and so on. As long as the set of data is small enough for a simple graph, we can take advantage of the visual data view. For large data sets, the graph display does not provide the previously mentioned visual advantage, but queries and manipulations of data are

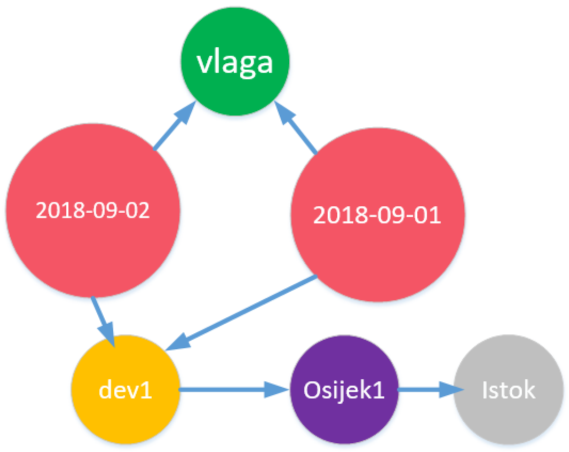

Figure 4. Result of Cypher query in the graph database

required, as in relational databases. The next chapter will show that a query of the graph database is much easier than a query of the relational database, as well as provide a visual display that can be easily interpreted.

V. Data querying

Retrieving data in the relational database was done using SQL queries. SQL (Structured Query Language) is a structured, standard language for accessing and manipulating databases [11]. In the data model for this paper, for example, the data that corresponds with the question "show all the moisture measurements of device dev1 in the region east" is to be found. A Standard SQL query by relational database for such parameters would be:

SELECT metrics.id, measurements.date, device.name, dimensions_type.type, region.name

FROM device, location, region, measurements, measurements_type

WHERE device.location = location.id and location.region = region.id and measurement.device = device.id and metering.type = measurements_type.id and region.name = 'east' and measurements_type.type = 'humidity' and device .name = 'dev1'

The result of this long query is a simple table containing the requested data and is shown in Table I.

TABLE I. Result of SQL query of relational database

id	***date***	***name***	***type***	***name***
1	1.9.2018	dev1	vlaga	istok
2	2.9.2018	dev1	vlaga	istok

In order to obtain the result of the query, it was necessary to read the related data of five tables which will be displayed in a single table for the sake of convenience.

The Cypher syntax is used for the same query in the graph database. Like SQL, Cypher is a declarative query language that allows users to specify which actions they want to perform (such as matching, inserting, updating or deleting) on graph data. Cypher is a platform-independent open code used on a variety of graph ecosystems. The syntax is in the ASCII format, which provides a well-known, legible way to collapse patterns of nodes and relationships within graph data sets [12].

Using the same question as on the test data model, "show all moisture measurement of device dev1 in the east region", a Cypher query was used on the data in the graph database:

MATCH
*p = (d: Device) - [*] -> () - [] -> (t: Type), w = (d: Device) - [: Below * 2]*

WHERE
d.name = "dev1" AND t.name = "moisture"

RETURN p, w

One can note that the Cypher syntax is much shorter than the SQL query, and is more understandable. As a result, we get the graph showing the required data shown in Figure 4. The figure immediately shows the relationship of data (nodes) and their values.

In the above example, the structure of the graph database followed the standard structure of the relational database where the data was divided into several tables and connected with foreign keys. If the data structure was adapted to the graph database then the requested query based on the graph database would be even simpler. When all the data is added to the nodes representing all the devices and all the measurements, they are mutually linked, as shown in Figure 5, where yellow nodes represent devices, and red-labeled nodes represent measurements. Device nodes have these defined properties: name, location, region; while measurement nodes have the following defined properties: date, value, type of measurement. In this case, another measurement feature has been added – the value of the measurement that was not used in the previous model. This is another advantage of the graph database, where certain nodes can add different, often unrelated properties that may need to be used in future analyses. Adding new properties to relational databases would require much more action than creating new tables and linking them to specific data from other tables.

With such a small data model it is very easy to visually notice the extremes – in this case, devices that have not made any measurements are clearly visible. Each node is defined with properties that can be displayed by clicking on a particular node shown in Figure 6. or in tabular view of the properties of a particular node.

For the same query used in the previous model, "show all moisture measurements of device dev1 in the east region", Cypher query was used according to the graph database:

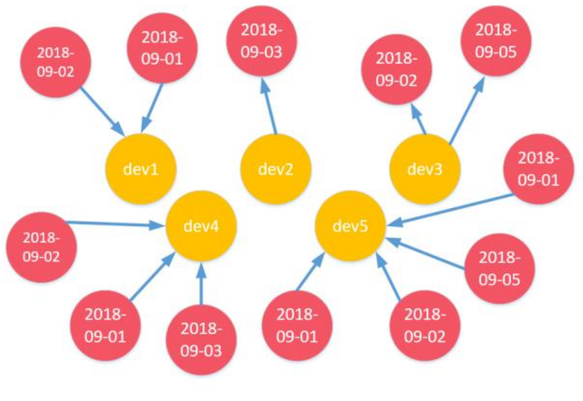

Figure 5. Graph Data model with two types of nodes

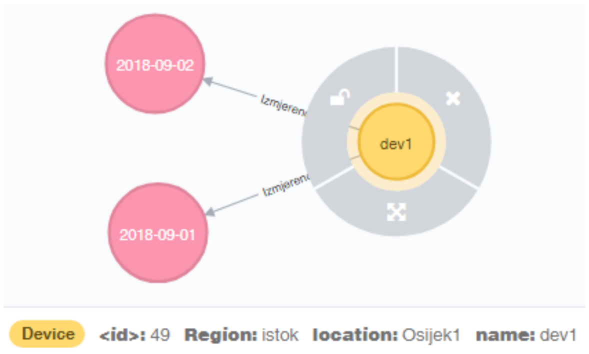

Figure 6. Result of cypher query used in a new data model

MATCH
p = (d: Device) - [*] -> (m: Measurements)

WHERE
d.name = "dev1" AND m.type = "moisture" AND d.Region = "east"

RETURN p

This shorter Cypher query gets the identical result as in the previous model shown in Figure 4. described in previous chapter IV.

For this data model, a shorter and more transparent Cypher query is required in relation to the SQL query based on the relational database. All data is read from a graphical view or can be read as shown in Table II.

TABLE II. TABULAR DISPLAY OF THE VALUE AND PROPERTIES OF THE REQUESTED CYPHER QUERY

"p"
[{"name":"dev1","Region":"istok", "location":"Osijek1"},{},{"date":"2018-09-01", "type":"vlaga","value":"100"}]
[{"name":"dev1","Region":"istok", "location":"Osijek1"},{},{"date":"2018-09-02", "type":"vlaga","value":"20"}]

VI. CONLUSION

Due to an increasing tendency of connecting devices through the communication network and the increasing number of generated data that needs to be stored and processed for future use and data analyses, the question of choosing a suitable database is raised. Data is still stored in relational databases because they are already in a structured form and can be easily stored, manipulated and analyzed in relational databases. By increasing the IoT devices, connecting devices of different characteristics, and obtaining different unstructured data that needs to be stored, a solution is proposed in the form of non-relational databases. This paper compares the use of a relational database (MySQL) with a non-relational database (Neo4j) on the test data model. Data manipulation compares data retrieval requirements in the form of SQL queries for the relational database and Cypher queries for the graph database. The query result was identical, but the Cypher query was better and more simply structured than the SQL queries. It is also possible to achieve a visual representation of the results from the query result layout which is graphically displayed in the form of nodes and interconnections in the graph database..The relational database displays the data in the form of a table, while, in the graph database, the result can be displayed both in the form of a table and in the form of a graph. According to the studied and available literature [12], [13] it can be concluded that the manipulation of unstructured, abundant data is simpler in the graph database environment. Although the test data model was well-structured and did not contain large amounts of data, one can note the advantage of using of a graph database for IoT systems that will, in the future, have more and more unstructured data that will need to be analyzed and processed as quickly and efficiently as possible.

In future work it would be useful to examine how easily large amounts of data can be manipulated and presented. For this purpose, the single programming approach of both databases should be used and the cases in which it is better to use a relational or non-relational database ought to be examined in a measurable way.

REFERENCES

[1] A. Stojanović, "Osvrt na NOSQL baze podataka - četiri osnovne tehnologije", Polytechnic & design, Vol. 4, No. 1, 2016.

[2] NoSQL or SQL? Do you have to choose?, [Online], https://blog.rackspace.com/nosql-or-sql-do-you-have-to-choose

[3] K.Rabuzin, M.Šestak, „Grafovske baze podataka – pregled istraživanja i budućih trendova", 41st international convention on information and communication technology, electronics and microelectronics, Opatija, 2018.

[4] IoT the Internet of Connected Things, [Online], http://www.grapheverywhere.com/portfolio-item/iot/

[5] A Graph Platform Reveals and Persists Connections , [Online], https://neo4j.com/product/#overview

[6] Graph databases: making meaning from the Internet of Things, [Online], https://www.information-age.com/graph-databases-making-meaning-internet-things-123458606/

[7] H. Smidt, M. Thornton, R. Ghorbani, „Smart Application Development for IoT Asset Management Using Graph Database Modeling and High-AvailabilityWeb Services", Proceedings of the 51st Hawaii International Conference on System Sciences, 2018.

[8] G.Tripathi, B.Sharma, S.Rajvanshi, „A Combination of Internet of Things (IoT) and Graph Database for Future Battlefield Systems", International Conference on Computing, Communication and Automation, 2017.

[9] A.Gupta, S.Tyagi, N.Panwar, S.Sachdeva, „NoSQL Databases: Critical Analysis and Comparison", International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), 2017.

[10] What is XAMPP?, [Online], https://www.apachefriends.org/index.html

[11] Introduction to SQL, [Online], https://www.w3schools.com/SQl/sql_intro.asp

[12] Cypher, The Graph Query Language, [Online], https://neo4j.com/Cypher-graph-query-language/

[13] C.Küçükkeçeci, A.Yazıcı, „Big Data Model Simulation on a Graph Database for Surveillance in Wireless Multimedia Sensor Networks", Big Data Res., 2017.