Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-120

Comparing Graph and Relational Database Management Systems for Querying Data Warehouses

Elena Milovanović, Ana Pajić
Faculty of Organizational Sciences, University of Belgrade
Belgrade, Serbia

***Abstract*— Businesses face the problem of processing extremely large amount of data every day. Finding and analyzing relationships between enormous set of connected data will be the key to successful business. Thus, our work discusses graph databases which are designed for dealing with densely connected data. The paper is focused on comparing Neo4j graph database and traditional Oracle relational database for querying data warehouses. The first results show that Neo4j graph database better deals with more complex questions when amount of data increases, which is very important for multidimensional analysis. Moreover, the query performance does not depend on graph dimensionality but only on size of subgraph covered by query.**

***Keywords*— big data; data warehouse; graph database; query performance; multidimensional analysis**

I. Introduction

In today's world where competition is very strong, getting better market position and reaching competitive advantage are demanding. Amount of data available is getting bigger and bigger every day and might be petabytes or exabyte of data consisting of billions to trillions of records from different sources. The term big data is used to describe "…datasets whose size is beyond the ability of traditional databases to capture, store, manage and analyze" [1].

The problem with loading and using data has shown up due to the huge amount of data. It can be solved using database sharding by achieving the spread load of data. Relational database (hereinafter RDBMS) is the dominant persistent storage technology and it implements ACID (Atomicity, Consistency, Isolation, Durability) set of transaction properties. RDBMS cannot easily support distinct distributed database servers that can process data independently due to its table based structure. This has led to development of another type of databases, named NoSQL (not only SQL) databases that can support smooth maintaining of huge amount of data. NoSQL databases do not distribute logical entity across multiple tables and they are stored in one single place. Referential integrity between logical entities is not demanding and they are trying to preserve consistency inside single entity, but often even this is not provided.

Graph databases will be briefly mentioned in this paper, since they can help finding relationships between enormous sets of data. Graphs are nodes connected by edges, where adding a new concept, genome, or relationship does not involve redesigning the whole database. As such, graphs form the basis for naturally representing genome and biological data [2]. Moreover, the application of graphs on DNA analysis and protein synthesis is presented in [3]. It is pointed out that "graphs truly are one of the most useful structures for modeling objects and interactions" [4]. Company's ability to understand, analyze and use big graphs of densely connected data will be a key of success when trying to take advantage over its competitors. Awareness of mentioned above, puts in a center of interest not only data but also connections between them.

To this end, this paper is focused on comparing Neo4j graph database and traditional RDBMS for querying data warehouses. The first results of our benchmark study show that Neo4j graph database performance does not drop off markedly as the amount of data increases. Moreover, the query performance does not depend on graph dimensionality but only on size of subgraph covered by query.

The remaining of this paper is structured as following. We first describe the basic concepts of Neo4j graph data model, its advantages and shortcomings in comparison to RDBMS. In section 3 possible usages of graph databases in data warehousing is discussed. Section 4 describes the implementation of our benchmark study and the most relevant results are given in section 5. Section 6 follows with a discussion of results and ideas for future work.

II. Graph Databases

A. *Differences Between Relational and Graph Databases*

NoSQL today is the term used to address to the class of databases that do not follow RDBMS principles, specifically being that of ACID nature, and are specifically designed to handle the speed and scaling of the likes of Google, Facebook, Yahoo, Twitter, and many more [5]. NoSQL systems can be grouped in many ways depending on criteria used. The most spread categorization is based on data model on which system is based. According to this criterion, NoSQL systems can be divided into four classes: Key-value stores, Document stores, Column family, Graph databases. This work is focused on graph databases.

Graph space can be very wide and complex. According to [6] there are two categories of graph space:

- Graph databases which enable online graph persistence and these are usually accessed from some application. They can be compared to OLTP databases.

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-120

- Graph compute engines which denote technologies that enable offline data processing. They are similar to technologies that can be used for data analyzing, like data mining and OLAP.

Among many advantages like fast data access, indexing and so on, the most important, that distinguish graph databases from relational, are better performances and schema-less structure. NoSQL databases are far much better in dealing with connected data than RDBMS. When amount of data is increasing performances of join queries are tremendously decreasing. In this situation NoSQL databases stay stable because their performance does not depend on data volume. Because of its structure and schema, graphs are by their nature upgradable and expendable. New subgraphs can easily be added to existing data set without affecting application functionalities or previous queries execution. When we are talking about schema depending on when it is verified there are two different approaches - schema on write and schema on read. This approach distinguishes graph and RDBMS. In traditional database management system (hereinafter DBMS), like relational, schema is read, verified and applied on data loading. This means that if data do not fit schema, load is failing and transaction rollbacks. On the other hand, NoSQL databases use approach schema on read. This means that schema is applied in a moment of reading data when user issues a query. This enables fast loads and supporting multiple schemas for the same data. Also data format might be unknown because queries against the data haven't been defined.

For problem presentation Neo4j technology will be used as representative of graph databases.

B. *Basic Concepts in Graph Data Model*

Nodes and relationships are basic building blocks in graphs. Node can be used to represent any entity in graph and it stores data that depict specific entity. In Neo4j node can contain attributes, relationships with other nodes and labels. Each relationship in graph has its specific start and/or end nodes, dangling relationships are not allowed. For relationship, type can be defined which can be treated as relationship name. Relationships can have attributes as well as nodes. Even though relationships are directed, they are equally well traversed in both directions so there is no need to create duplicate relationships in the opposite direction. In Neo4j relationship of one node with itself is also supported, in other words relationship can have the same start and end node.

Attributes are presented as key-value pairs, where key is defined as string and values can be defined as some predefined data type or array which elements are predefined types. Null values are not allowed, if value is missing then that attribute will be omitted. Labels are used for node grouping which means that all nodes that has the same label, belong to same set. Labels are simplifying queries because it is possible to run query on certain labeled subset in graph. We will be free to compare the effect of labels to effect of columns that we use in SQL GROUP BY statement in relationship databases. One node can have zero or many labels. Label name must not be an empty string. Since labels can be attached and detached in runtime, they can also be used for denoting of node current state. Path contains one or more connected nodes. Paths are usually a result of query execution on graph database or a result of traversals. Fig. 1 shows the logical structure of Neo4j graph database data model. In this meta-model all concepts and relationships between them are presented. For meta-model presentation IDEF1X notation is used.

III. Graph Databases in Data Warehousing

In today's world data warehousing is very important aspect for decision making process and future planning. In order to this we can use OLAP operations like roll-up, drill-down, slice-and-dice and pivot on multidimensional data cube model. Even though graphs are studied for a long period of time, weak point for graph databases still is natural OLAP support. For multidimensional analysis in RDBMS there are many algorithms and systems that can support them. This area has not been explored that much in multidimensional networks. In RDBMS e.g. we can calculate amount of products sold in each store, each city, or in entire country but this kind of analysis on multidimensional networks will not give only numbers but also network structure for each product, city, and country as well. This can lead us to interesting results and maybe will help us understand why sale is higher in certain store, city or country. Using demographic and other data in graph we could also see purchasing habits, standard of living for some country and get wider picture, that will help us understand resulting numbers.

In past few years there are more and more papers that deal with using graphs in data warehousing. In [7] Graph Cube is introduced, which is a model that can be used in data warehousing. It can provide OLAP queries on large multidimensional networks. For its implementation characteristics of multidimensional networks in combination with existing data cube techniques are used.

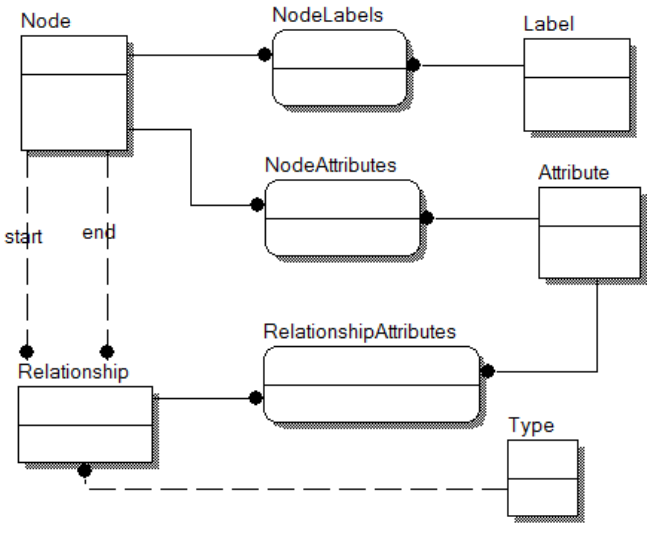

Figure 1. Neo4j graph database meta-model

This new concept enables using aggregations and SQL GROUP BY statement from RDBMS world on multidimensional data cube model. Graph Cube integrates both multidimensional attributes and network structures.

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-120

That paper introduced a new kind of query, which crosses multiple multidimensional spaces of the network and these are called cross-cuboid queries (crossboid queries). This means that aggregate networks can become the measure of a graph cube.

Bachman [8] in his work speaks about structural analytics which is an additional category of analytical processing which deals with qualitative and quantitative aspects of the actual structure of the graph network. Business Intelligence with Integrated Instance Graphs (BIIIG) framework for integration and business intelligence based on graph data is proposed in [9], [10]. This framework uses graph models to present metadata and instance data from different sources, like Enterprise Resource Planning (ERP) and other similar systems. In [11], the new graph semantic that is based on object oriented multidimensional data model, named GOOMD, is presented. In order to specify conceptual level design of data warehouse, this model defines a list of graph based formal constructs. GOOMD gives a realization of different logical structures of data warehouse like star, snowflake, and so on. Also this model provides an operational model for OLAP that is used for operations like roll-up, drill-down, slicing, dicing, drill-across and drill-through.

There are some technologies that are free and can be used in combination with some graph database system. For instance, RapidGrapher product can be used for data integration. It provides a simple way to map data from existing relational database into a single, searchable and navigable graph, and thus allows managing RDBMS data as a graph. Bachman in his work introduced GraphAware Neo4j framework. It offers Neo4j server extension that allows developers to build (REST) application program interface (API) on top of Neo4j using Spring Model–view–controller (MVC). Also, available is a runtime environment for both embedded and server deployments which allows the use of pre-built and custom modules, which author called GraphAware Runtime Modules. These modules extend the core functionality of the database by transparently enriching, modifying and preventing ongoing transactions in real-time.

Since Neo4j Cypher already provides some analytical features for graph pattern matching and aggregation, they will be used for benchmark study.

IV. Solution Implementation

This research utilizes a test dataset provided free by Microsoft- the AdventureWorks. The dataset is divided into four schemas: Sales, Production, Purchasing, Human resources and one common model called Person. For our purpose, we were concerned with querying on the sales related facts within the dataset. Tables OrderHeader, OrderDetail, Product, Customer and Territory is used for the benchmark study. The original data values provided by Microsoft the AdventureWorks dataset is used since its size is large enough to effectively compare query performance. In this dataset 19 820 customers, 10 territories, 504 products and 31 465 orders are stored. Fig. 2 shows a selection of tables and their relationships using IDEF1X notation.

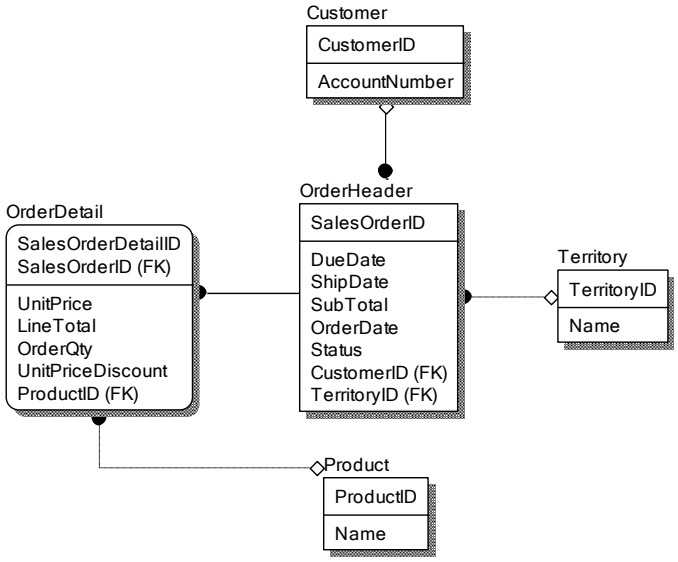

Figure 2. Relational schema

As RDBMS Oracle 11g Express Edition is used. Stored data is then converted into CSV files and imported into Neo4j 2.2.1 environment. For each tuple in each table, one node is created and the relationships between all nodes. For this example there are 31 465 relationships for Order – Customer, 31 465 relationships for Order – Territory, 54 998 for Order – OrderDetail and 31 998 for OrderDetail – Product, resulting in 149 926 relationships at all.

While OLAP query performance comparison between the traditional RDBMS and Neo4j graph database is a primary objective of this research, the control environment is used and both systems are installed on the exact same machine hardware. The data systems were not running simultaneously while the other was being queried and all other unnecessary processes on the machine were shutdown to allow maximal resource utilization in testing phase.

Neo4j has its own language Cypher which is declarative and it uses ASCII-Art to represent patterns. This language enables to describe what user wants to select, insert, update or delete from Neo4j. Also it is used for creating nodes, labels, relationships and properties. Since relationships are equally well traversed in both directions there's no need to create duplicate relationships in the opposite direction. In this example four relationships are created:

- Territory – ORDERED_IN -> Order
- Customer – BOUGHT -> Order
- Order – HAS -> OrderDetail
- Product – PRODUCT_DETAIL -> OrderDetail

Fig. 3 presents node Order with number 43677 and its relationships. Order has three types of relationships and these are: HAS with node OrderDetail, ORDERED_IN with Territory and BOUGHT with node Customer. The orientation of these relationships can be seen on the Fig. 3. Fourth relationship that is created for this experiment is PRODUCT_DETAIL between nodes OrderDetail and Product. Each node has its attributes according to relational data model that was presented earlier.

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-120

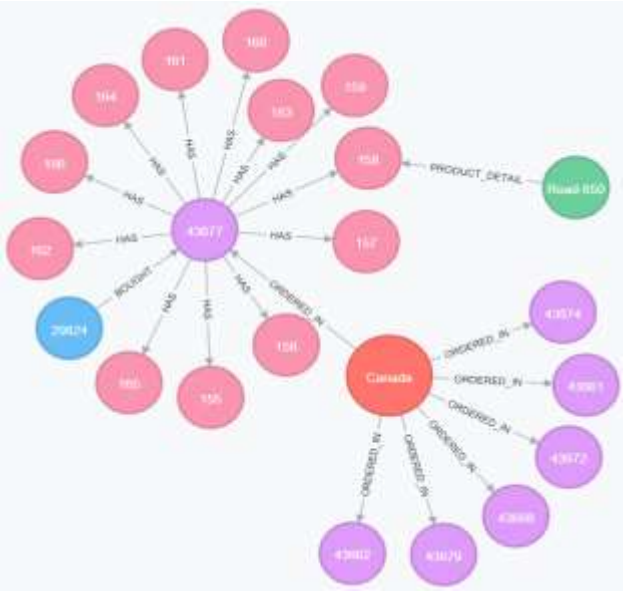

Figure 3. Node Order and its relationships

v. Experimental Results and Analysis

Aggregate functions are particularly important for using graphs in data warehousing. To calculate aggregated data, Cypher offers aggregation, much like SQL GROUP BY statement. Aggregate functions take multiple input values and calculate an aggregated value from them. Example is AVG function that calculates the average of multiple numeric values, or MIN function that finds the smallest numeric value in a set of values. Aggregation can be done over all the matching subgraphs, or it can be further divided by introducing key values. These are non-aggregate expressions that are used to group the values going into the aggregate functions.

To test the performance of using Neo4j in data warehousing, queries were designed with one or more logical joins to simulate questions complexity that business users might ask in regards to this dataset. We executed the queries on different amount of data on both DBMS in order to show Neo4j performance stability. Three different queries are executed on the dataset and they are presented in Table I. In order to quantify the query performance, we measured multiple execution times for our earlier established queries. Ten execution times are measured and mean execution time is presented. Mean execution time for query Q1, expressed in milliseconds, is presented in Table II.

If we compare execution time for Neo4j and Oracle measured on 5000 records, we can see that Neo4j needs noticeably more time. In next two phases of experiment, same query has been executed on 10000 and 19119 records, respectively.

TABLE I. THREE TYPES OF QUERIES

	Cypher	SQL
Q1	MATCH (c: Customer)- [p] -> (o: Order) RETURN c, sum (o.SubTotal) order by c.CustomerID	select c.customerid, sum(oh.SUBTOTAL) from order_header oh join customer c on (c.customerid = oh.customerid) group by c.customerid order by c.customerid
Q2	MATCH (o: Order)-[r] -> (d: orderDetail) RETURN o, avg (d.LineTotal) order by o.SalesOrderID	select oh.salesorderid, avg(od.linetotal) from order_header oh join order_detail od on (oh.salesorderid = od. salesorderid) group by oh.salesorderid order by oh.salesorderid
Q3	MATCH (t:Territory) -[w]-> (o:Order) -[q]-> (d:orderDetail) <-[r]- (p:Product) return t.Name as territory , p. Name as product, sum (d.OrderQty) order by t.Name, p.Name	select t.name as territory, p.name as product, sum (od.ORDERQTY) from order_header oh join order_detail od on (oh.salesorderid = od.salesorderid) join territory t on (oh.territoryid = t.territoryid) join product p on (od.productid=p.productid) group by rollup(t.name, p.name) order by t.name, p.name

TABLE II. Q1 QUERY EXECUTION RESULTS

Amount of data	Neo4j mean execution time (ms)	Percentage increase of mean execution time (Neo4j)	Oracle mean execution time (ms)	Percentage increase of mean execution time (Oracle)
5000	0.71590		0.08240	
10000	0.83650	16,85 %	0.15180	84,22 %
19119	1.08840	52,03 %	0.29470	257,65 %

Percentage increase of mean execution time between each phase of experiment in comparison to first phase is also presented in the table. Thus we get much better, comprehensive review. Now we can notice that there was a significant increase in execution time for Oracle RDBMS results as number of records grows. Neo4j has proven to be much more stable when the amount of data increases. These results support the assumption that NoSQL databases stay stable because their performance does not depend on data volume. On Fig. 4 plot of percentage increase in execution time is presented. X axis refers to amount of data and Y axis refers to percentage increase in execution time. Red line is used for Oracle results and blue line for Neo4j results.

Results for query Q2 have also been measured in ten iterations and mean execution time, expressed in milliseconds, is presented in Table III.

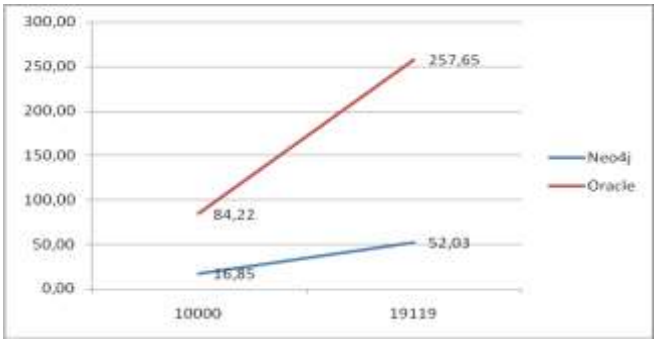

Figure 4. Q1 query- plot of percentage increase in execution time

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-120

TABLE III. Q2 QUERY EXECUTION RESULTS

Amount of data	Neo4j mean execution time (ms)	Percentage increase of mean execution time (Neo4j)	Oracle mean execution time (ms)	Percentage increase of mean execution time (Oracle)
5000	0,6780 ms		0,1334 ms	
10000	0,9343 ms	37,80 %	0,1990 ms	49,18 %
11600	1,2231 ms	80,40 %	0,2385 ms	78,79 %

We can notice the similar situation on Fig. 5. The interesting thing is that percentage increase in mean execution time for third phase, with 11 600 records, is similar for both databases. More precisely, Neo4j results greater increase in comparison to Q1 query execution results. This could be explained by number of relationships that should be traversed between nodes Order and OrderDetail. Previously, we have shown that 54 998 relationships have been created between these two types of nodes. On the other hand, 31 465 relationships were formed between nodes Customer and Order. This can explain greater mean execution time since in second query there are almost twice as many relationships that should be traversed then in the first query. In this case, size of subgraph affected execution time. The second assumption has been proved - execution does not depend on graph dimensionality but only on size of subgraph covered with query.

The third query was design with two logical joins to simulate questions complexity. Results for Q3 have also been measured in ten iterations and presented in Table IV. It is well known that performance of join queries is tremendously decreasing in RDBMS, when amount of data increases. Our experiment shows that the mean execution time for third phase, with the most records, is similar for both databases. The Neo4j graph database better deals with more complex questions which is very important for multidimensional analysis. As shown in Fig. 6, there is a significant increase in mean execution time for Oracle RDBMS than for Neo4j database as number of records grows.

Given the results of the queries performance, we can conclude that these results are suggestive of the fact that graph databases are good solution for dealing with big data. Neo4j database is better solution when the amount of data increases.

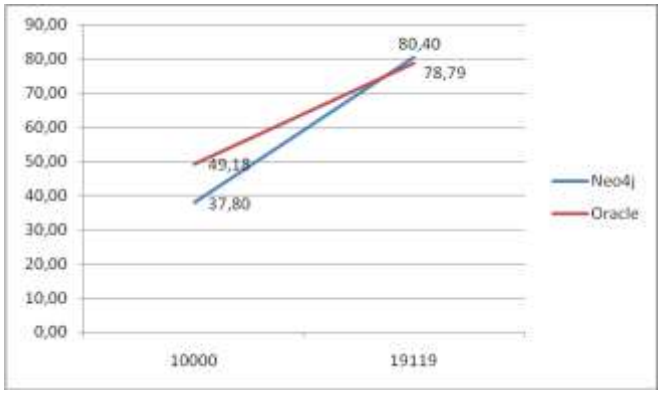

Figure 5. Q2 query- plot of percentage increase in execution time

TABLE IV. Q3 QUERY EXECUTION RESULTS

Amount of data	Neo4j mean execution time (ms)	Percentage increase of mean execution time (Neo4j)	Oracle mean execution time (ms)	Percentage increase of mean execution time (Oracle)
100	0,8947 ms		0,1078 ms	
300	0,8738 ms	-2,34	0,1274 ms	18,18
438	0,9467 ms	8,34	0,1584 ms	24,33

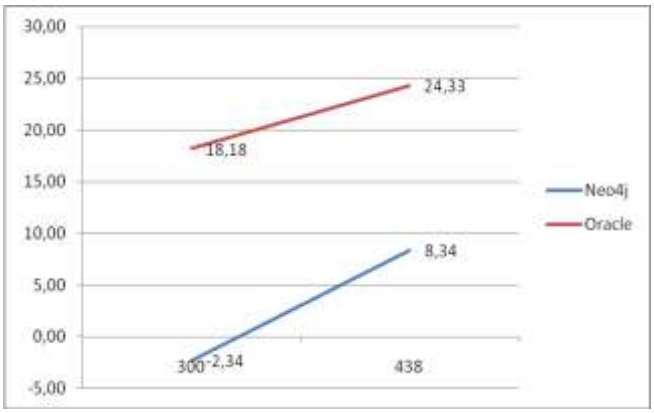

Figure 6. Q3 query- plot of percentage increase in execution time

In all cases, the percentage increase in mean execution time for Neo4j database is significantly smaller than for Oracle RDBMS, when amount of data increases. Different amount of data in each case is due to the fact that queries in graph databases do not access all data but just a certain part of graph.

VI. Conclusion

For every company getting better market position and reaching competitive advantage are demanding. This can be achieved by using big data potential to help them improve operations and make faster, more intelligent decisions. Since this is not the problem RDBMS can handle with, new type of databases has shown up, known as NoSQL databases. In this paper, we studied a special type of NoSQL databases, namely Graph database. Special attention was paid to the possibility of using OLAP analysis in Graph databases.

We conducted several performance experiments to compare Neo4j graph database to Oracle relational database system, for use as underlying technology for multidimensional analysis. Neo4j graph uses its declarative language Cypher to cover basic analysis. It offers aggregation, much like SQL GROUP BY statement and has some group functions like MAX, MIN, SUM, AVG and some statistical analysis like standard deviation. The query performance was monitored on both systems and Neo4j had proven to be much more stable when the amount of data increases. In all cases, Neo4j needed noticeably more time, but there was a significant percentage increase of mean execution time for Oracle RDBMS than for Neo4j database as number of records grows.

Future studies can examine the performance of query execution on a larger dataset to better show the differences between mean execution time for Oracle RDBMS and graph database Neo4j. Moreover, we should explore the possibilities of using Cypher language for much more

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-120

complex data analysis. More attention will be paid on Graph cube as a solution for creating data cube model, since authors of Graph Cube did not consider to use Cypher for analyzing. In order to get better results, RapidGrapher could be used for data integration and GraphAware framework for OLAP analysis over created graph cube. We will extend our research to these new technologies and test them for data integration and in-depth OLAP analysis.

The problem of temporal graphs still lies unaddressed. Temporal graphs capture changes in graphs over time and efficient techniques to represent, store and query dynamic graphs are essential for using graph databases in data warehousing. Neo4j does not provide intrinsic support for time-varying graph, but the property graph representation can be used. It requires more complex data model and to associate nodes and edges with time intervals in arbitrary ways. In the future work we plan to investigate Neo4j graph and Cypher query language for versioning.

References

[1] S. A. Tinkhede and S. P. Deshpande, "Big Data- The Vast Growing Technology with its Challenges and Solutions," Int. J. Comput. Sci. Mob. Appl., vol. 3, no. 1, pp. 33–38, 2015.

[2] M. Graves, E. R. Bergeman, and C. B. Lawrence, "A graph conceptual model for developing Human Genome Center databases," Comput. Biol. Med., vol. 26, no. 3, pp. 183–197, 1996.

[3] S. Navlakha, M. C. Schatz, and C. Kingsford, "Revealing biological modules via graph summarization," J. Comput. Biol. a J. Comput. Mol. cell Biol., vol. 16, no. 2, pp. 253–264, 2009.

[4] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, "A comparison of a graph database and a relational database," in Proceedings of the 48th Annual Southeast Regional Conference, 2010, pp. 1–488.

[5] G. Vaish, Getting started with NoSQL. Packt Publishing, 2013.

[6] I. Robinson, J. Webber, and E. Eifrem, Graph Databases. 2013.

[7] P. Zhao, X. Li, D. Xin, and J. Han, "Graph Cube: On Warehousing and OLAP Multidimensional Networks," in Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, 2011, pp. 853–864.

[8] M. Bachman, "GraphAware : Towards Online Analytical Processing in Graph Databases," Imperial College London, 2013.

[9] A. Petermann, M. Junghanns, R. Muller, and E. Rahm, "BIIIG: Enabling business intelligence with integrated instance graphs," in 5th International Workshop on Graph Data Management (GDM 2014), 2014, pp. 4–11.

[10] A. Petermann, M. Junghanns, R. Müller, and E. Rahm, "Graph-based Data Integration and Business Intelligence with BIIIG," in Proceedings of the VLDB Endowment, 2014, pp. 1577–1580.

[11] A. Sarkar, S. Choudhury, N. Chaki, and B. Swapan, "Conceptual Level Design of Object Oriented Data Warehouse: Graph Semantic Based Model," J. Comput. Sci., vol. 8, no. 4, pp. 60–70, 2009.

About Author (s):

Elena Milovanović is a teaching assistant of Information systems at Faculty of Organizational Sciences, University of Belgrade. Her current research interests include diffrent aspects of Information systems design, Geographic information systems and Database Management Systems.

Ana Pajić is a teaching assistant of Information systems at Faculty of Organizational Sciences, University of Belgrade. Her current research interests include diffrent aspects of Business Information Systems, Semantic-based enterprise modeling and Database Management Systems.