Graph Databases Benchmarking on the Italian Business Register

Nicola Ferro[1] and Luca Sinico[2]

[1] Department of Information Engineering, University of Padua, Italy
ferro@dei.unipd.it
[2] Infocamere, Italy
luca.sinico@infocamere.it

Abstract. In this paper, we develop a benchmark for graph database systems based on the real data of the Italian Business Register, consisting of about 10 million entities and 5 million relationships among them.
We evaluate three state-of-the-art open source graph database systems – ArangoDB, Neo4J, and OrientDB – and we compare them to a well-known relational database management system, namely PostgreSQL.
We found out that the strong points of graph databases are: the purposely designed storage techniques, which let them have good performance on graph datasets; and the purposely designed query languages, which go beyond the standard SQL and manage the typical problems that arise when graphs are explored. However, we have seen that the main performance increments have been obtained when heavy graph situations are queried; for simpler situations and queries, a relational database performs equally well.

1 Introduction

Graph databases [3, 2] are becoming a more and more adopted data model and technology. Common use cases for graph databases are social graphs, recommender systems, business relationships, network impact analysis, geospatial applications such as maps and route planning for rail or logistics, telecommunication or energy distribution networks, fraud detection, and many more.

The success of graph databases in such fields is due not only to their data model, which naturally model the domain of interest, but also to their query functionalities and their graph processing APIs which allow us to express queries and operations on graphs more easily than, e.g., the SQL language.

In this paper, we focus on the application of graph databases to the case of the Italian Business Register[3] ("Registro Imprese"), i.e. the register of company details (name, articles of association, directors, headquarters, ...), their share holders, their subsidiaries, and all subsequent events that occurred after registration – for instance, amendments to the articles of association and company roles, relocations, liquidations, insolvency procedures, and so on.

[3] http://www.registroimprese.it/

The Italian Business Register thus provides a complete picture of the legal position of each company and is a key archive for drawing up indicators of economic and business development. Its main function is to ensure an organic system of legal disclosure for companies, ensuring the provision of timely information throughout the country. The Italian Business Register is thus intrinsically a graph of relationship among Italian companies. This work has been conducted in the context of a joint project with InfoCamere s.c.p.a., the company which manages the Italian Business Register. We aim at comparing graph database systems, also with respect to traditional relational systems, to determine the best solutions to develop an innovative application to search and access the Italian Business Register. Therefore, we studied and tested three of the most important open source graph databases – ArangoDB[4], Neo4j[5], OrientDB[6] – and we also compared them to a well-known relational database, namely PostgreSQL[7]. We used a real dataset, i.e. the data of the Italian companies and their equity participations, constituted by, roughly, 6 million companies, 4 million persons and 5 million relationships among them.

The paper is organized as follows: Section 2 briefly summarizes related works; Section 3 describes the compared graph databases; Section 4 introduces the domain of interest and the experimental setup; Section 5 reports the conducted experiments and their results; finally, Section 6 wraps up the discussion and outlooks possible future works.

2 Related Works

In recent years there have been several efforts to evaluate graph databases and develop benchmarks for this purpose. [10] compared Neo4J to the MySQL[8] relational database management system by using synthetic data. [5] used synthetic data to evaluate traversal operations for different graph database systems, among which Neo4J, OrientDB, and DEX (now Sparksee, commercial)[9]. [7] used both synthetic data and real data from the Amazon's co-purchasing network to evaluate several systems, among which Neo4J, OrientDB, DEX/Sparksee, and InfiniteGraph[10] (commercial). [6] used synthetic data to evaluate Neo4J, OrientDB, and DEX/Sparksee according to different workloads – load, traversal, and intensive. [8] compared a wide range of graph and relational database systems using synthetic data against specific test cases, namely Single Source Shortest Paths (SSSP) problem, Shiloach-Vishkin connected components algorithm, and PageRank. [1] used two real datasets, namely Wiki-Talk and Slashdot, to evaluate Neo4J, OrientDB and other systems. Finally, [9] used the *Linked Data*

[4] `https://www.arangodb.com/`
[5] `https://neo4j.com/`
[6] `https://orientdb.com/`
[7] `https://postgresql.org/`
[8] `https://www.mysql.com/`
[9] `http://sparsity-technologies.com/`
[10] `http://www.objectivity.com/products/infinitegraph/`

Benchmark Council (LDBC)[11] *Social Network Benchmak* (SNB) to evaluate Neo4J, PostgreSQL and other systems. Note that LDBC develops also another benchmark, i.e. the *Semantic Publishing Benchmark* (SPB).

Our work differs from the state-of-the-art because it relies on real data, instead of synthetic ones as many of the works above, and explores a new domain, i.e. the business register, which is not covered by previous works.

3 Property Graph Databases

A *property graph* [4] is a directed graph were both nodes and edges can contain *properties* which, typically are, key-value pairs. Moreover, both nodes and edges are typed, allowing for semantic enrichment of the data.

We compared the graph databases described in the following sections. Table 1 summarizes and compares the main features of the analyzed graph databases.

ArangoDB ArangoDB is an open-source NoSQL multi-model database management system supporting three data models, namely graphs, key/value pairs, and documents. It is a schema-free DBMS, implemented in C/C++ and JavaScript, and supports several operating systems (Linux, OS X, Windows, Raspbian and Solaris).

ArangoDB represents documents in a proprietary *JavaScript Object Notation (JSON)* binary format, called *VelocyPack*, which is a platform-independent serialization format. It manages two kinds of documents, i.e. nodes and edges. It has its own proprietary and declarative query language – *ArangoDB Query Language (AQL)*, which offers aggregate, ordering, filtering, and sub-querying, and graph-traversal functions, especially suitable to express how graphs have to be explored, e.g. breadth or depth-first, how many times to visit a node or an edge, whether to avoid duplicates and cycles.

Neo4j Neo4j is an open-source NoSQL graph database implemented in Java and Scala, portable across many operating systems. Its data model consists of node objects, which can be labeled, connected by named and directed edge objects, where both nodes and edges can act as containers of properties. Neo4j stores graph data in different files for each part of the graph – nodes, relationships, labels, and properties – specifically arranged to facilitate graph traversals.

Cypher is the proprietary and declarative query language provided by Neo4j. It is designed for working with graph data and defines the structure of the patterns to be searched or created over the graph data. It allows for a sort of graphical specification of queries or search patterns by means of an ASCII art like drawing.

[11] `http://www.ldbcouncil.org/`

Feature	ArangoDB	Neo4j	OrientDB	PostgreSQL
Category	NoSQL	NoSQL	NoSQL	Relational
Initial release	2012	2007	2010	1996
Database Model	Graph Document Key-Value	Graph	Graph Document Key-Value Object	Relational Object
Graph model	Property graph	Property graph	Property graph	–
Native graph	Yes	Yes	Yes	–
Index-free adjacency	No	Yes	∼Yes	–
Implementation	C++ JavaScript	Java Scala	Java	C
Indices	Yes, secondary	Yes, secondary	Yes, secondary	Yes, secondary
Transactions (Single instance)	Yes, ACID	Yes, ACID	Yes, ACID	Yes, ACID
Data scheme	Schema-less	Schema-less	Schema-less Schema-ful	Schema-ful
Referential integrity	∼Yes (edges)	Yes (edges)	∼Yes (edges)	Yes
Data typing	Yes	Yes	Yes	Yes
Query Language	AQL	Cypher	SQL “extended”	SQL
Stored procedures	AQL JavaScript	Java	SQL JavaScript Groovy	PL/pgSQL PL/Tcl PL/Perl PL/Python
Graph functions	Yes	Yes	Yes	No
Drivers	JavaScript Java PHP Python Perl .Net ...	Java C/C++ JavaScript PHP Ruby Python ...	Java JDBC JavaScript PHP Ruby Python ...	C/C++ JDBC PHP Ruby Python ODBC ...
Access methods	RESTful HTTP	RESTful HTTP Java API	RESTful HTTP Java API Binary	JDBC C API
Triggers	∼Yes (via FOXX Queues)	∼Yes (via Event Handler)	∼Yes (Hooks)	Yes
Concurrency	Yes	Yes	Yes	Yes
User concepts	Yes	Yes	Yes	Yes
Durability	Yes	Yes	Yes	Yes
Community license	Apache v2	GPL v3	Apache v2	BSD
Replication conflict resolution	Master/Master Master/Agent	Master/Slave	Master/Master Master/Slave	Master/Slave
Data sharding	Yes	No	Yes	No
Caching	Data Query results	Data Query plans	Data Query results	Data Query plans

Table 1. Feature matrix

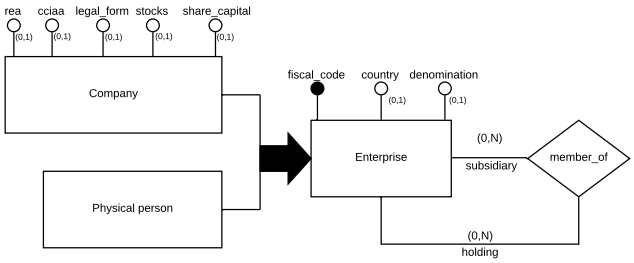


Fig. 1. ER schema of the simplified Italian Business Register.

OrientDB OrientDB, initially born as an object-oriented database, is now an open-source multi-model database management system, supporting alternative paradigms, namely objects, documents, key/value pairs, and graphs. It is developed in Java and multi-platform. It relies on its object-oriented nature to model graphs as node and edge elements together with their properties.

OrientDB comes with a sort of dialect of the SQL query language, which provides specific extension for graph traversal and pattern specification.

4 Experimental Setup

4.1 Use Case

InfoCamere[12] is the IT company of the Italian Chambers of Commerce. By developing up-to-date and innovative IT solutions and services, it connects the Chambers of Commerce and their databases through a network that is also accessible to the public via the Internet. Thanks to InfoCamere, businesses, Public Authorities, trade associations, professional bodies and simple citizens both in Italy and abroad can easily access updated and official information and economic data on all businesses registered and operating in Italy. The Italian Chambers of Commerce are public bodies entrusted to serve Italian businesses through over 300 branch offices located throughout the country. InfoCamere helps them in pursuing their goals in the interest of the business community, especially through the provision of the Italian Business Register and services over it.

[12] `https://www.infocamere.it/`

We use a simplified version of the Italian Business Registers, whose *Entity–Relationship (ER)* schema is shown in Figure 1. There are two entities, namely *companies* and *physical persons*, whose attributes are:

- `fiscal_code`: a unique identifier;
- `rea`: an identification code for linking to the REA register (Repertorio Economico Amministrativo);
- `cciaa`: the identifier of the Chamber of Commerce where the company is registered;
- `type`: either company or physical person;
- `denomination`: the business name for companies or the name for persons;
- `country`: the registration country for companies, e.g. foreign companies with branches in Italy or the birth country for persons;
- `legal_form`: a code representing the legal nature of a company, e.g. sole proprietorship;
- `share_capital`: the amount of company's share capital;
- `stocks`: the number of financial stocks.

The recursive relationship distinguishes between two roles: the `subsidiary` and the `holding` company.

The dataset is a snapshot of the Italian Business Register at October 2016, consisting of about 10.5 million companies and physical persons and about 5 million relationships among them.

4.2 Queries

According to the requirements for developing advanced services on the Italian Business Registry, there are two typologies of queries that need to be answered, e.g. for supporting antitrust, investigations, financial organizations, loans and so on:

- *how is a company composed?*
  - retrieve the companies which own equity shares of a given company, i.e. the *holding companies*;
  - retrieve the companies directly connected to a given company regardless of type of participation, i.e. both *subsidiaries* and *holdings*;
  - retrieve the list and level of all the direct and indirect subsidiaries of a given company, e.g. shell companies;
- *how are companies linked?*
  - retrieve the list of the companies which are shared subsidiaries of two given companies;
  - retrieve the list of the companies which are shared holdings of two given companies;
  - get the shortest path between two given companies.

Each query is performed on three types of workloads: *small-weight* (also referenced as Small), *medium-weight* (Medium), and *large-weight* (Large) workload for that query. For example, for the "Subsidiaries and holdings of a company" query, the light situation is given by a company that has few (units) subsidiaries and holdings; the medium case is given by a company with a good number (hundreds) of subsidiaries and holdings; the heavy case is given by a company that has a very large number (thousands) of subsidiaries and holdings.

We used the query language provided by each database to implement the queries: AQL for ArangoDB; Cypher or the Java API in the case of complex queries for Neo4j; its extended SQL for OrientDB; and SQL for PostgreSQL.

4.3 Configuration

All tests have been done during an internship at InfoCamere. We set up a virtual instance of a RedHat Linux server with the following characteristics: AMD Opteron 6276, x86_64, dual-core, single-thread per core, 64bit, 2.3GHz; 6GB DRAM; 100GB HDD; Red Hat Enterprise Linux Server release 6.8 (Santiago); OpenJDK 1.8.0_101 64-Bit Server VM (Java version 8u101).

We used the following versions of the tested databases: ArangoDB 3.0.10; Neo4J 3.0.6; OrientDB 2.2.11; and, PostgreSQL 9.6.1.

We performed a cache warm-up; we repeated each query 20 times, reporting averaged values and confidence intervals; after each query, we re-started the database and warmed-up the cache again.

5 Evaluation

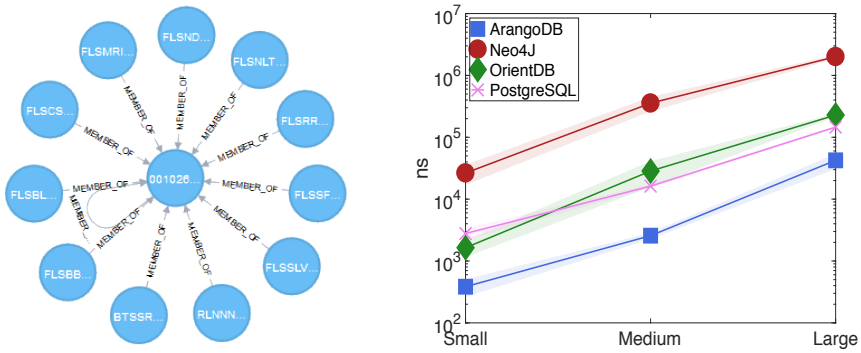
5.1 How is a Company Composed?

In this section we report the performance of the queries corresponding to the use cases aimed at defining how a company is composed.

Holdings of a company Figure 2 shows, on the left, the typical subgraph for the holdings of a company and, on the right, the performance of the different systems in the three workloads, where we have 4 resulting nodes in the small case, 418 nodes in the medium case, and 6067 nodes in the large case.

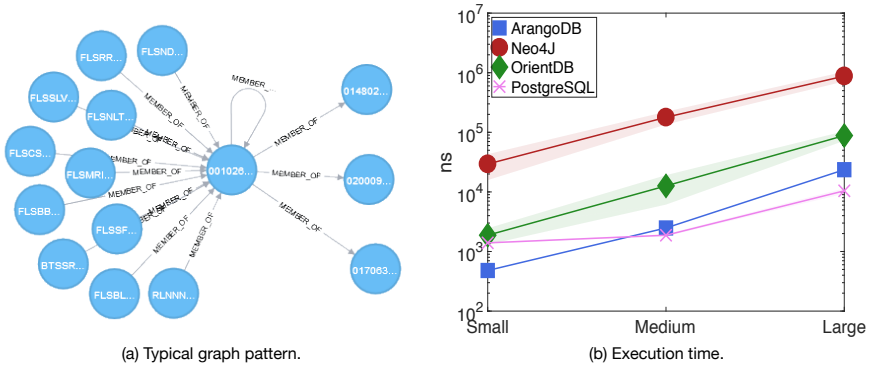
From Figure 2.(b), it emerges that the best performing system is ArangoDB while the worst one is Neo4J; OrientDB and PostgreSQL are very similar and about one order of magnitude slower than ArangoDB. In terms of stability, i.e. low variance among query performance, PostgreSQL is the top performing system, closely followed by ArangoDB.

Subsidiaries and holdings of a company Figure 3 shows, on the left, the typical subgraph for the subsidiaries and holdings of a company and, on the right, the performance of the different systems in the three workloads, where we



(a) Typical graph pattern. (b) Execution time.

Fig. 2. Holdings of a company.



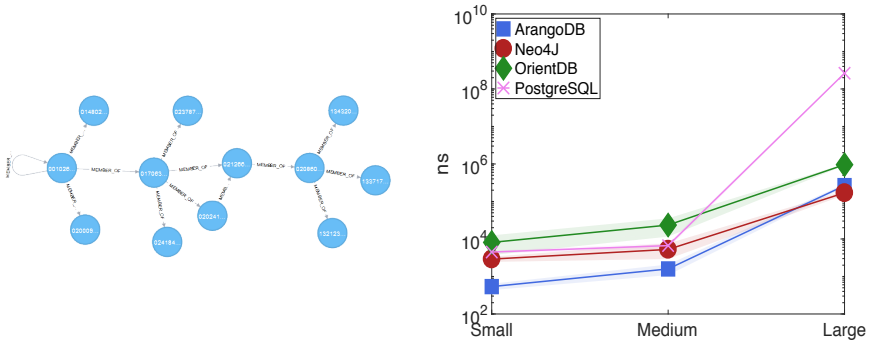
(a) Typical graph pattern. (b) Execution time.

Fig. 3. Subsidiaries and holdings of a company.

have 5 resulting nodes in the small case, 120 nodes in the medium case, and 1815 nodes in the large case.

As before, ArangoDB is the top performing system while Neo4J is the worst one. However, PostgreSQL is very competitive and better than ArangoDB in the medium and large cases. OrientDB performs roughly an order of magnitude slower than ArangoDB. In terms of variance, both ArangoDB and PostgreSQL are quite stable while OrientDB and Neo4J exhibit quite less stability.

Direct and indirect subsidiaries of a company Figure 4 shows, on the left, the typical subgraph for the direct and indirect subsidiaries of a company and, on the right, the performance of the different systems in the three workloads, where we have 12 resulting nodes in the small case, 76 nodes in the medium case, and 14037 nodes in the large case. Note that a challenging aspect of this query is a kind of "global uniqueness", i.e. the need to avoid duplicate records and to traverse the same nodes more than once.



(a) Typical graph pattern. (b) Execution time.

Fig. 4. Direct and indirect subsidiaries of a company.

We can observe that ArangoDB is still the top performing systems but the gap between systems is somehow smaller in this more complex case, being just about one order of magnitude for the small and medium cases. In these two cases, PostgreSQL is still very competitive with respect to graph databases while its performance greatly deteriorates in the large case, about three orders of magnitude slower. This is probably due to the way in which "global uniqueness" is imposed, i.e. only at a posteriori and not while traversing the graph. From the variance point of view, PostgreSQL is the most stable system closely followed by ArangoDB while, as in the previous cases, Neo4J and OrientDB are somehow less stable.

5.2 How are Companies Linked?

In this section we report the performance of the queries corresponding to the use cases aimed at defining how companies are linked among them.

Companies which are shared subsidiaries of two companies Figure 5 shows, on the left, the typical subgraph for the shared subsidiaries of two companies and, on the right, the performance of the different systems in the three workloads, where we have 6 resulting nodes in the small case, 84 nodes in the medium case, and 8728 nodes in the large case.

ArangoDB is again the top performing systems and the gap between systems is somehow small, being just about one order of magnitude for the small and medium cases. In these two cases, PostgreSQL is still very competitive with respect to graph databases while its performance greatly deteriorates in the large case, about four orders of magnitude slower. As before, from the variance point of view, PostgreSQL is the most stable system closely followed by ArangoDB while Neo4J and OrientDB are somehow less stable.

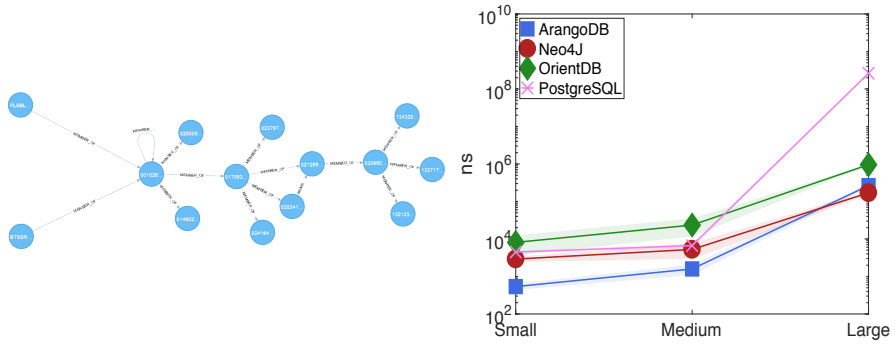


Fig. 5. Companies which are shared subsidiaries of two companies.

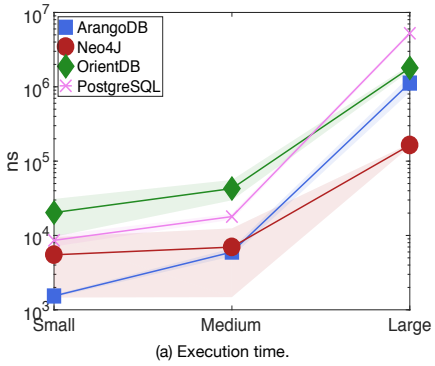
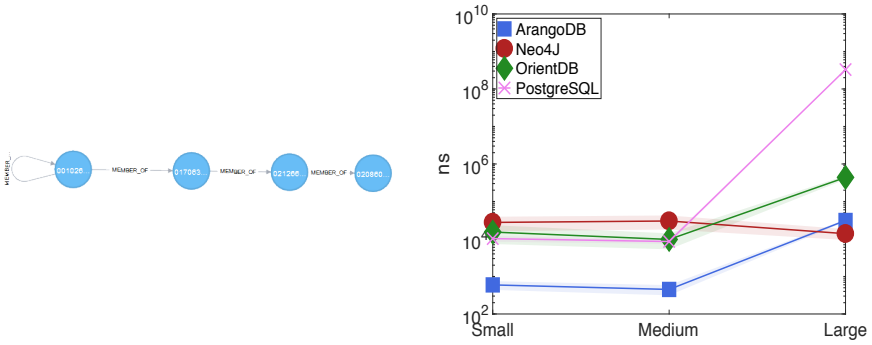


Fig. 6. Companies which are shared holdings of two companies. The typical graph pattern is the same as in Figure 5.(a) but with reversed edges.

Companies which are shared holdings of two companies Figure 6 shows, on the left, the typical subgraph for the shared holdings of two companies and, on the right, the performance of the different systems in the three workloads, where we have 26 resulting nodes in the small case, 140 nodes in the medium case, and 19548 nodes in the large case.

This query is basically the same as the previous one but just with reversed direction of the edges. With respect to the previous query case, we can observe that there is a swap between ArangoDB, the top performing system in the small and medium cases, and Neo4j in the large case, where Neo4J is the best system. Moreover, in the small and medium cases, the variance in performance is greatly increased for both Neo4J and OrientDB. PostgreSQL somehow performs as in the previous query case, even if it improves in the large case being just about one order of magnitude slower.



(a) Typical graph pattern. (b) Execution time.

Fig. 7. Shortest path between two companies.

Shortest path between two companies Figure 7 shows, on the left, the typical subgraph for the shortest path between two companies and, on the right, the performance of the different systems in the three workloads, where we have 5 resulting nodes in the small case, 120 nodes in the medium case, and 1815 nodes in the large case.

ArangoDB is again the top performing systems and the gap between systems is somehow small, being just about one order of magnitude for the small and medium cases. In these two cases, PostgreSQL is also very competitive with respect to graph databases while its performance greatly deteriorates in the large case, about three orders of magnitude slower. It is interesting to note that ArangoDB and Neo4J have very close performance in the large case and that the Neo4J performances are almost constant across the three cases. Finally, from the variance point of view, PostgreSQL is the most stable system closely followed by ArangoDB while Neo4J and OrientDB are somehow less stable.

5.3 Overall Considerations

By looking at the charts, it emerges that ArangoDB generally performs better than all the others, especially for the small and medium cases. For the large case, Neo4j works on par with ArangoDB or even better in all the queries that requires somehow heavier graph processing.

Furthermore, Neo4j and OrientDB have often quite close performance, especially for small and medium cases; for the large case, instead, there is no clear winner.

Overall, PostgreSQL typically performs well and it is competitive in the small and medium cases or when light graph processing is required, i.e. when just directly connected nodes are involved. On the other hand, it takes more time than the others when it has to process bigger amounts of data and/or more complex graph structures.

6 Conclusions and Future Work

We evaluated the performance of three state-of-the-art open source graph database systems – ArangoDB, Neo4J, and OrientDB – and one relational database management system – PostgreSQL. We developed a benchmark using the real data of the Italian Business Register, considering six types of queries involving more or less complex graph patterns, and accounting for three workloads – light, medium, and large. To the best of our knowledge, this is the first benchmark for graph databases using real business register data.

We found that ArangoDB is almost always the top-performing system, especially in the small and medium cases, followed by Neo4J and OrientDB for which there is no clear winner. When we consider the large workload case, ArangoDB and Neo4J get closer and, sometimes, the latter performs better. When it comes to PostgreSQL it is competitive for the small and medium cases but its performance deteriorate in the case of large workloads and more complex graph structures. In terms of performance stability, PostgreSQL is the most stable system, i.e. the one with lowest variance, closely followed by ArangoDB while Neo4J and OrientDB show a greater variance.

Future work will be about the release to the Chambers of Commerce and other users the application which uses the graph database technology, and will further inspect how to extract other valuable information from the graph data.

References

1. Abul-Basher, Z., Chignell, M.H., Godfrey, P., Yakovets, N.: TGDB: Towards a Benchmark for Graph Databases. In CASCON 2016. pp. 257–267. ACM Press, New York, USA (2016)
2. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoč, D.: Foundations of Modern Query Languages for Graph Databases. ACM CSUR 50(5), 68:1–68:40 (2017)
3. Angles, R., Gutierrez, C.: Survey of Graph Database Models. ACM CSUR 40(1), 1:1–1:39 (2008)
4. Angles, R., Gutierrez, C.: An introduction to Graph Data Management. arXiv.org, Databases (cs.DB) arXiv:1801.00036 (2017)
5. Ciglan, M., Averbuch, A., Hluchy, L.: Benchmarking Traversal Operations over Graph Databases. In ICDEW 2012. pp. 186–189. IEEE Computer Society (2012)
6. Jouili, S., Vansteenberghe, V.: An empirical comparison of graph databases. In: SocialCom 2013. pp. 708–7015. IEEE Computer Society (2013)
7. Kolomičenko, V., Svoboda, M., Holubová, I.: Experimental Comparison of Graph Databases. In: IIWAS 2013. pp. 115–124. ACM Press, New York, USA (2013)
8. McColl, R., Ediger, D., Poovey, J., Campbell, D., Bader, D.A.: A Performance Evaluation of Open Source Graph Databases. In: PPAA 2014. pp. 11–17. ACM Press (2014)
9. Pacaci, A., Zhou, A., Lin, J., Tamer Özsu, M.: Do We Need Specialized Graph Databases?: Benchmarking Real-Time Social Networking Applications. In GRADES 2017. pp. 12:1–12:7. ACM Press (2017)
10. Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., Wilkins, D.: A Comparison of a Graph Database and a Relational Database. In: ACM SE 2010. pp. 42:1–42:6. ACM Press (2010)