

Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB

Diogo Fernandes¹ and Jorge Bernardino^{1,2}

¹*Polytechnic of Coimbra - ISEC, Rua Pedro Nunes, Coimbra, Portugal*

²*CISUC – Centre for Informatics and Systems of the University of Coimbra, Portugal*
a21220608@isec.pt, jorge@isec.pt

Keywords: Graph databases, NoSQL databases, AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, OrientDB.

Abstract: Graph databases are a very powerful solution for storing and searching for data designed for data rich in relationships, such as Facebook and Twitter. With data multiplication and data type diversity there has been a need to create new storage and analysis platforms that structure irregular data with a flexible schema, maintaining a high level of performance and ensuring data scalability effectively, which is a problem that relational databases cannot handle. In this paper, we analyse the most popular graph databases: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J and OrientDB. We study the most important features for a complete and effective application, such as flexible schema, query language, sharding and scalability.

1 INTRODUCTION

Nowadays, data volume is growing exponentially in social networks, such as Facebook and Twitter, which store and process massive amounts of data daily, where this data reaches petabytes of storage. Current relational databases are predominant in the market, but their adaptability is poor in processing connected data. Actual platforms must deal with huge amounts of data and related information. For this reason, a special type of databases designated by graph databases have appeared.

A graph database is a non-relational database that provides an effective and efficient solution for the storage of information in current scenarios, where data is increasingly interconnected. The storage mechanisms of graph databases are optimized for graphing, for the way they store adjacent records linked by direct references. In this adjacency list, each vertex maintains references to their adjacent vertices, forming an index species for the vertices on neighbourhood. This property is known as index-free adjacency (Robinson et al., 2015). Interest in graph models has been increasing in recent years due to their applications in areas like Semantic Web and Social Network Analysis (Dietrich et al., 2008). This type of database is easy to understand because its concept is based on the theory of graphs. This theory is basically based on graphs, which are mathematical structures used to

model relationships between objects. In this context, a graph is basically a structure, which is represented by nodes, or also called vertices (the entities), by edges (the relations) that are the lines that connect the various nodes and by properties (attributes). Therefore, the graph databases can simply be described as a way of representing and storing data, using their structures: nodes, edges, and properties. For that reason, graph databases are optimized for storing and querying graphs.

The problem with graph databases is that they are not particularly efficient in all desired operations such as in the representation of data that are derived from relational models. Therefore, they do not replace relational databases but are in fact an efficient solution when dealing with huge volumes of data that contain many related data.

The focus of this paper is the study of the main characteristics, advantages and use cases of graph databases. We study five of the most popular graph databases: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J and OrientDB.

The rest of this paper is structured as follows. Section 2 describes the advantages and uses of graph databases. Sections 3 to 7 describe AllegroGraph, ArangoDB, Infinite Graph, Neo4J and OrientDB, respectively. Section 8 presents the comparison of the five graph databases. Finally, Section 9 presents the main conclusions and future work.

2 MAIN ADVANTAGES AND USES OF GRAPH DATABASES

Graph databases are getting popular as data increases in scalability and relationships become the first choice over the relational databases. Ironically, legacy relational database management systems (RDBMS) are poor at handling relationships between data points. Their tabular data models and rigid schemas make it difficult to add new or different kinds of connections. Graphs are the future. Not only do graph databases effectively store the relationships between data points, but they are also flexible in adding new kinds of relationships or adapting a data model to new business requirements (Webber et al, 2015). Graph databases did not see a greater advantage over relational databases until recent years, when frequent schema changes, managing enormous volume of data, real-time query response time and more intelligent data activation requirements, make people realize the advantages of the graph model.

This technology is disrupting many areas, such as supply chain management, e-commerce recommendations, security, fraud detection and many other areas in advanced data analytics.

The main advantages of a graph database are the following (Guia et al., 2017) and (Neo4J, 2018):

- Better optimization on gathering information than relational databases;
- ACID rules support;
- Support of the data storage in the order of petabytes (10^{15});
- Allow for new types of data;
- Easily expandable due to its structural model - graph model;
- Suitable for complex and irregular data, normally involved in real world;
- Excellent for data mining operations;
- A high performance while querying deep data when compared to relational databases;
- There is no need to declare the data type for the nodes or edges, unlike relational model;
- They are very agile in development, since they can be easily adapted over time;
- Combination of multiple dimensions to manage big data, including time series, geo-dimensions, and hierarchies on different dimensions.

In short, the three key advantages are: performance, flexibility, and agility.

Performance, because with traditional databases relationship queries will come to a grinding halt as the number and depth of relationships increase. In

contrast, graph database performance stays constant even as our data grows year after year.

Flexibility, because IT and data architect teams move at the speed of business because the structure and schema of a graph model adjusts itself as applications and industries change. Rather than exhaustively modeling a domain ahead of time, data teams can add to the existing graph structure without endangering current functionality.

Finally, agility because developing with graph databases aligns perfectly with today's agile, test-driven development practices, allowing graph database to evolve in step with the rest of the application and any changing business requirements.

However, as any new technology replacing old technology, there are still obstacles in adopting graph databases. One is that there are fewer qualified developers in the job market than the SQL developers. Another is the non-standardization of the graph database query language (Wu, 2017).

In the next sections, we analyze five of the most popular graph databases.

3 AllegroGraph

AllegroGraph is a modern, consistent and persistent Resource Description Framework (RDF) graph database with high performance that is currently in use in open source, commercial, and US Department of Defense projects. AllegroGraph is characterized by the efficient use of memory by combining disk storage, making it possible to scale up to one billion nodes, always maintaining top performance. Basically, it provides services including vision building, rapid prototyping and proof-of-concept development, complete enterprise technology solution stack, and best practices to maximize value from semantic technologies.

AllegroGraph provides an architecture through the REST protocol, which is an architectural style that consists of a coordinated set of constraints applied to components, connectors, and data elements within a distributed system. Developers have developed adapters for the various supported languages, Sesame Java, Sesame Jena, Python, using proprietary Sesame and Lisp signatures.

AllegroGraph's competitive advantages are the following (Ranking of Graph DBMS, 2018):

- Suited to support *ad hoc* queries through SPARQL, Prolog and languages like JavaScript;
- Sorted quintuple indices that will index every primary and non-primary field so users never

have to worry about whether a certain field is indexed or not;

- It is possible to mix Geospatial, Temporal, Social Network Analytics, and Reasoning, all in the same query (SPARQL or Prolog);
- Triple Level Security with Security Filters;
- Gruff - Graph Visualization, Generate SPARQL and Prolog queries visually;
- Full and Fast Recoverability;
- Online Backups, Point-in-Time Recovery, Replication, Warm Standby;
- SOLR and MongoDB Integration;
- All clients based on REST Protocol;
- Cloud-Hosted AllegroGraph - Amazon EC2.

One of AllegroGraph limitations is it focuses on geo-temporal reasoning and social network analysis.

Figure 1 shows an example of Allegro Graph interface. In the middle, we have a graph database and on the left we have subtitles for the nodes and edges colours.

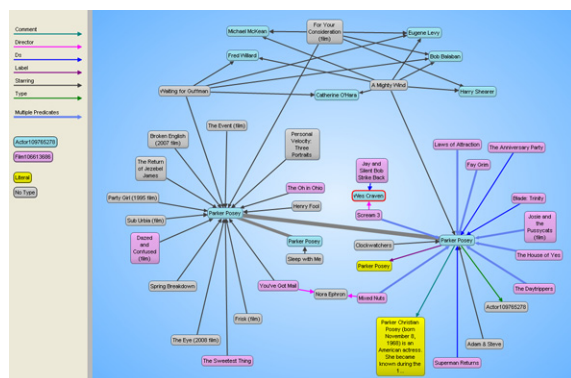


Figure 1: AllegroGraph interface. (AllegroGraph interface, no date).

4 ArangoDB

ArangoDB is a multi-model database system developed by triAGENS GmbH. The data can be stored as key or value pairs, documents or graphs and all of this can be accessed by just one query language (AQL - ArangoDB Query Language). ArangoDB uses the same core and same query language for all the data models. If a new product is being developed and then new ideas are generated, the model requires necessary changes frequently and ArangoDB provides the opportunity of keeping the product updated. Combining different data models in one query makes data less complicated. This multi-model makes sense because it has simplified performance scaling, increased flexibility, fault

tolerance, huge amount of storage memory and lower cost than other databases. This software has two versions: Community and Enterprise editions.

The ArangoDB cluster architecture is a master / master CP (Consistent and Partition) (Mehra, 2017) model without a single point of failure. With "CP", we mean that in the presence of a network partition, ArangoDB prefers to maintain its internal consistency instead of availability. Clients will have the same view of the database regardless of the node to which they connect, and the cluster continues to fulfil requests even when a machine fails. In this way, ArangoDB was designed as a distributed multi-model database (ArangoDB, no date).

The most important advantages and features of ArangoDB are:

- Management of multiple data models with a single core and query language (AQL);
- HTTP API to manage database;
- Multi architecture – unique instance, cluster or mixed services;
- JavaScript Foxx Framework integration;
- Sharding;
- Cloud ready on Azure Cloud;
- Consolidation - minimizes the components, reducing the complexity of the technology stack. This means a lower total cost of ownership, increasing flexibility and consolidating technical needs;
- Simplified Performance Scaling - can easily react to growing performance and storage needs by independently scaling with different data models. ArangoDB scales both vertically and horizontally, and if our performance needs decrease, we can easily scale down the back-end system to save on hardware and operational requirements;
- Reduced Operational Complexity - Polyglot Persistence concept is about choosing the right data model for the right job. A native multi-model database allows to have polyglot data without the complexity, but with data consistency on a fault tolerant system;
- Strong Data Consistency - a single back-end manages different data models with support for ACID transactions. It provides strong consistency on a single instance and atomic operations when operating in cluster mode;
- Fault Tolerance – it uses a multi-model database and a consolidated technology stack. By design, ArangoDB enables modular architectures with different data models running and works for cluster usage as well.

It was created in 2012, but the community is still small. Also, the software complexity may require an experienced user on database operations despite all available documentation.

ArangoDB has a web interface as illustrated in Figure 2. It shows an example of a graph database with the node labels on the top right corner. On the left there is a menu with options like cluster statistics, graph databases collections, query editor, user information and application services.

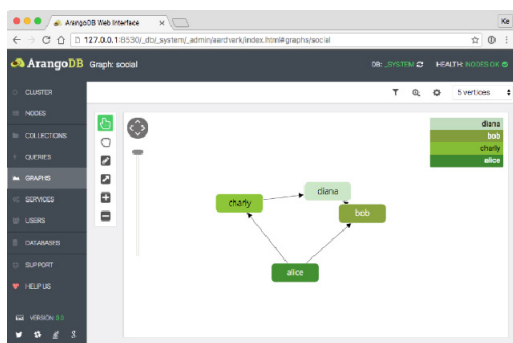


Figure 2: ArangoDB interface. (source: docs.arangodb.com)

5 InfiniteGraph

InfiniteGraph is a distributed graph database implemented in Java with core in C++ developed by Objectivity. InfiniteGraph is unique in their massive scalability, distributed ingest and processing capabilities and has cloud storage, among other features. It only offers physical disk storage. InfiniteGraph has grown at a tremendous pace, and can be found in areas such as network management and telecommunications, healthcare, cyber security, crime prevention, predictive analytics and fraud detection, bioinformatics, genomics, scientific research, finance, social CRM applications supporting enterprise sales and marketing, and social networking (InfiniteGraph, no date). The physical model is object-oriented, so two classes are used to store the vertices and nodes of a base, BaseVertex and BaseEdge, respectively.

InfiniteGraph is built on a highly scalable, distributed database where data and processing are distributed. A single graph database can be partitioned and distributed across multiple disks and multiple machines with the ability to query data between these frontiers. The same database can access the graph database locally or over a network. The lock server handles the locking requirements

through database applications, allowing for simultaneous reading and writing access to the graph database. Access to the database is not controlled when a database instance is created, but rather at the transaction level. Data servers handle remote databases and application requests for distributed grid databases and nodes. Although the REST interface facilitates interactive access to a database from a browser, it is not suitable for high-performance access because it deals with many large graphs.

Following, InfiniteGraph key features and advantages are described:

- API/Protocols: Java (core C++);
- Multi-property-graph model;
- Online backups;
- Multi-threading processing;
- Cloud enabled;
- Friendly graph visualization tool;
- Parallel query support;
- Export data models to JSON and GraphML;
- Distributed - scalable graph database solution that distributes the processing load in the most efficient way for applications;
- Cost Effective - reduces the total cost of ownership by reducing the need for data movement and transformations.

InfiniteGraph also has some limitations. It does not support sharding and does not have a free version.

In Figure 3, we have an example of the InfiniteGraph interface.

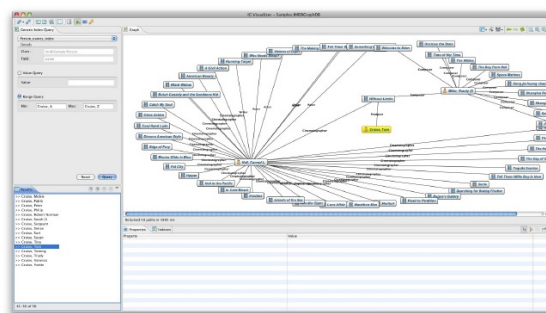


Figure 3: InfiniteGraph interface.

6 Neo4J

Neo4J is an open-source graph database implemented in Java. The developers describe Neo4J as a fully transactional database and a persistent Java engine where we can store structures in the form of graphs instead of tables. There are many connections inside

a big data and Neo4J helps find the intertwining links between them with the help of its visually interactive graphs. Neo4J uses natives graph storage which provides the freedom to manage and store data in a highly disciplined manner. Neo4J is considered the most popular and used graph database worldwide, used in areas such as health, government, automotive production, military area and others, being a major reference in this area.

This software was created in 2007 and is divided into three broad versions: a Community Edition that is a free version, an Enterprise Edition, where there is a possibility to test a more complete version for 30 days, and there is still a Government Edition, which is like an upgrade to Enterprise version. This version is highly focused for government services. The main differences between the two major versions of Neo4j are: an existence of online backups, high cache memory performance, detailed monitoring of the system, strong management of database locks, and greater scalability of the database, among other benefits of the Enterprise Edition (Neo4J, no date).

Neo4J typical application scenarios are real-time recommendations, identity and access management, network and IT operations, fraud detection, anti-money laundering / tax evasion, knowledge graphs, graph analytics and algorithms, graph-powered artificial intelligence, smart homes and IoT.

Currently most databases run through a server, which can be accessed through a client library. The Neo4j can run in the embedded mode, as well as in server mode. Embedded mode must be understood as an instance that stores all the information on the disk, and not as an in-memory database, that works exclusively with the internal memory of the machine. The embedded Neo4j is ideal for hardware devices, desktop applications, and for embedded applications in servers. Running Neo4j in server mode guarantees a more common way to implement a database, which is what is most used today. On each server there is an embedded instance of Neo4j.

Neo4J has many competitive advantages, which makes this software one of the most used ones in this area. Following, we describe the major features of Neo4J:

- Flexible schema;
- Follows property graph data model;
- Scalability and reliability;
- Cypher query language;
- HTTP API to manage the database;
- Index support using Apache Lucence;
- Drivers support like Java, Spring, Scala, JavaScript;
- Online backups;

- Cloud enabled;
- Exporting of query data to JSON and XLS format;
- Most active graph community in the world;
- High performance thanks to native graph storage and processing;
- Easy to learn and to use;
- Easy to load data into the software;
- Whiteboard-friendly data modelling to simplify the development cycle.

Neo4J doesn't support sharding and the Community version, being a free version, it has some limitations on the number of nodes, relationships and properties.

In Figure 4 we can see an example of the Neo4J interface running on a web browser. The sidebar on the left shows database information, node labels, relationship types and property keys. In the centre we have an example of a graph with multiple nodes (vertices), the relationships (edges) and attributes names, and on top we have text section where we can query data in Cypher language.

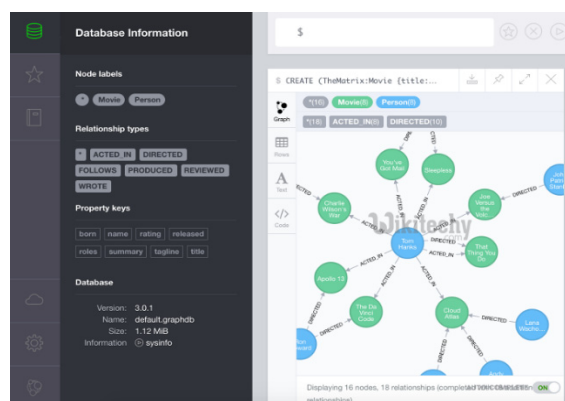


Figure 4: Neo4J interface.

7 OrientDB

OrientDB is a multi-model open source NoSQL database management system that supports data models in documents, graphs, key/value, and objects.

It was launched commercially in 2011 by OrientTechnologies and implemented in Java. It is transactional and supports distributed architecture with replication. The manipulation of the database can be done in Java, SQL or with Gremlin. Physical data storage can be done in memory and on disk.

Like all systems, it uses the free adjacency list to enable native query processing, but unlike the

others, it uses document database and object-orientation capabilities to store physical vertices. OrientDB supports schema less, full and mixed modes including SQL as one of the languages used and replication and sharding that can be used in most complex use cases. OrientDB provides safety in all confidential data that is present with the use of authentication, password and data-at-rest encryption.

It has a Community Edition version that is free (Apache 2 License) but does not support features such as horizontal scalability, fault tolerance, clustering, sharding and replication. Enterprise Edition is an extension and supports features such as query profiles, distributed cluster configuration, metric record, live monitoring, a migration tool and alert configuration. OrientDB use cases are fraud detection, network/IT operations, graph search, recommendation engines, master data management, identity management and forensic analysis.

The OrientDB architecture occurs in distributed mode. This can be distributed across different servers and used in diverse ways to achieve maximum performance, scalability and robustness.

OrientDB uses the Hazelcast Open Source project for automatic node discovery, storing runtime cluster configuration, and synchronizing certain operations between nodes.

Following, we present the key features and advantages of OrientDB:

- Supports SQL language;
- Web technologies support – HTTP, RESTful protocol, JSON libraries;
- Distributed – multi-master replication support;
- Cloud ready;
- Database manipulation using Java;
- It can embed documents like any other document database but also supports relationships;
- Multi-master plus sharded architecture, providing horizontal scalability and reliability;
- Fast installation;
- Free version using Apache Licence 2.

OrientDB have some limitations such as not having an import tool. Sometimes the documentation is outdated and some users have experienced some bugs with the graph editor.

OrientDB also provide a web interface. Figure 5 shows an example of a graph database with a sidebar bar, which shows nodes and edges properties, and some edition functions like add vertices or clear canvas on top right corner.

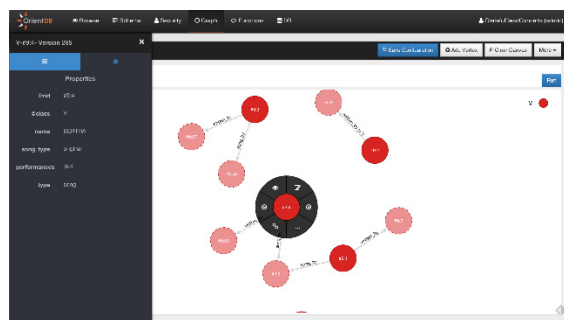


Figure 5: OrientDB interface. (OrientDB, no date).

8 COMPARISON OF FEATURES

After analysing the five graph databases and once the functionalities are the most important properties when choosing a graph database, a summary was made to represent the quality of each feature in each graph database. Based on (Buerli, 2012) and (Cox, 2017) we select the most important features that a graph database should have:

- Flexible schema - While relational databases require new tables or alterations in the existing ones to add new types of data, in a graph database we can add new type of vertices and edges without alterations in the previously stored data;
- Query language – Relational databases use SQL query language to manipulate the database, but graph databases needed a more powerful query language. There are many different query languages to manipulate specific graph databases like Cypher on Neo4J or AQL on ArangoDB, and we will study to see which one suits best the needs of querying data;
- Sharding - One of the key elements to being able to scale NOSQL databases is sharding, where a large dataset can be broken up and distributed across a number of (typically replicated) shards;
- Backups – Graph databases should provide functions for planning, performing and restoring a database backup. A full backup contains all data files and information required to restore a repository to the state it was in at the time of the backup;
- Multi-model – A multi-model graph database provides a database with unstructured data, and we can visualize relationships with data like in the form of graphs, key-value pairs, documents or tables;

- Multi-architecture - When planning a graph database solution, there are several structural decisions to make. These decisions may vary slightly, depending on which database product you choose. We will analyse which graph databases give the best options for the architecture to be implemented;
- Scalability - As hardware continues to innovate at a rapid pace (mass storage devices, CPU and networks), this in turn leads to increased capacity for existing database software, enabling growth in the number of transactions per second. There are two approaches to scaling a database: Vertical (Scale Up) and Horizontal (Scale Out). Vertical scaling involves adding more physical or virtual resources to the underlying server hosting the database – more CPU, more memory or more storage. Horizontal scaling involves adding more instances/nodes of the database to deal with increased workload;
- Cloud ready – A graph database implemented on cloud is a great feature because it solves scalability problems and provides real-time management.

Table 1 summarizes our study, where each row represents the most important features to take into consideration when choosing a graph database.

In this comparison a five Likert scale from 0 to 4 was used. The grade 4 means that the feature is well implemented in the software and 1 means that the feature is not well implemented and should be improved. 0 (zero) is assigned if the feature is not supported by the software. The grades are assigned according to our experience and literature review. Following, we present the legend for comparison: Great: 4 points; Good: 3 points; Average/Normal: 2 points; Bad: 1 point; Does not support: 0 points.

Table 1: Graph databases features comparison.

	AllegroGraph	ArangoDB	InfiniteGraph	Neo4J	OrientDB
Flexible Schema	1	3	3	4	3
Query Language	3	3	3	4	3
Sharding	3	3	0	0	3
Backups	3	2	3	4	3
Multimodel	4	4	2	2	4
Multi Architecture	3	4	3	4	3
Scalability	3	4	3	4	3
Cloud Ready	3	3	4	4	3
Total	23	26	21	26	25

The analysis of AllegroGraph demonstrates that this software has good features. However, comparing to the other graph databases, some features are worst implemented on AllegroGraph. It

also has a architecture supported by REST protocol and a query language named SPARQL. It is a language designed for querying RDF graphs but does not seem to be widely adopted on "SQL of Graphs". It is used more widely for querying larger data sets than closed local datasets. One of the main weaknesses is that the cloud support is still in beta version and does not provide a flexible schema.

ArangoDB has more functionalities than AllegroGraph. It provides the user the chance to use a flexible schema or not; on the other hand it is a schema-free database and thus allows flexible storage. All documents in a collection can have the same or totally different structures. ArangoDB exploit the similarities in document structures to save storage space. It will detect identical document schemas and will only save each unique schema once. This process is called shaping in ArangoDB. ArangoDB has a SQL like, powerful and modern language named AQL that enables really impressive complicated queries and processes to be pushed to the backend, which makes this a big strength. It also has functionalities like sharding, scalability, which is a multi-model database, and runs in the AzureCloud. The information available on ArangoDB backups is a bit confusing, with the community asking many questions, so it's a feature that they should improve.

Next, we have InfiniteGraph with functionalities like flexible schema, cloud ready, a well implemented backup tool, and it uses Gremlin language for querying data, a popular language for people working with graph databases. On the other hand, this software does not support sharding and uses only graph data models.

Neo4J is one of the best options when choosing a graph database and this study shows that this software has the most important features like flexible schema, a powerful query language named Cypher, a tool for planning, performing and restoring backups, two types of architecture (server mode and embedded mode), scalability and cloud prepared. Neo4J does not support sharding, and that is the main weakness of this software.

Finally, we have OrientDB. This software can work in a schema-less mode but also supports both schema-full and schema-hybrid solutions. OrientDB uses Java, SQL and Gremlin to manipulate data, supports sharding, can be implement in the cloud and has great scalability once it uses the distributed architecture. We could not find clear information on how the backups work.

After this analysis, we can conclude that Neo4J and ArangoDB stand out for their functionalities with 26 points, and is the best graph databases

option nowadays. We can use Neo4J that is optimized for graph databases and ArangoDB and OrientDB for databases with different data models (graphs, documents, key-value). However, we recommend that Neo4J is a better software with enhanced implemented features over ArangoDB.

Some websites like db-engines (Ranking of Graph DBMS, 2018), that study databases ranking according to their popularity, and Predictive Analytics Today (Top Graph Databases, 2017), that study trending graph databases, also have in their analysis Neo4J, ArangoDB and OrientDB on the top of their charts with Neo4J as the first choice in both. This makes this graph database the most used one in the world.

9 CONCLUSIONS AND FUTURE WORK

Graph databases provide more performance, flexibility and agility than non-relational databases. In this work, we analysed five of the most commonly used graph databases: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J and OrientDB.

We conclude that although they all have the same purpose, which is storing large volumes of data with many relationships between them, they are very different in the functionalities they offer. For example, Neo4J can be implemented in embedded mode or server mode, and the OrientDB can only be implemented in distributed mode. These five graph databases have some distinct advantages and features, but since the purpose of the software is the same, there are also many similarities.

When choosing a software to serve specific needs, we will have to see what each software offers, and graph databases are no exception. The functionalities that each one offers is what makes one better than the other and so an analysis was made for the features of these graph databases.

We conclude that Neo4J and ArangoDB offer the best functionalities to implement a graph database with Neo4J for standing out for its simplicity and due to its powerful query language named Cypher despite the need to have previous knowledge of it to manipulate the database. It also offers two architecture options, an intuitive interface and a flexible schema that allows the user to access very specific information very fast. ArangoDB is also a powerful tool, has an easy-to-learn and powerful query language (AQL), and a flexible schema. Furthermore, since it is a multi-model graph

database, it provides a lot of flexibility and supports sharding. These are two advantages over Neo4J and this is why it is also a good choice.

As future work, we intend to analyse the two best graph databases of this study - Neo4J and ArangoDB - in a real environment, using a benchmark to evaluate performance.

REFERENCES

- AllegroGraph interface, <http://franz.com/ps/newsletter-archive/32.png>
- AllegroGraph, <https://franz.com/agraph/allegrograph/>
- ArangoDB, <https://www.arangodb.com/>
- Buerli, M., 2012, "The Current State of Graph Databases", Department of Computer Science, *Cal Poly San Luis Obispo, California*.
- Cox, G., 2017, "Introduction to Graph Databases", <https://www.compose.com/articles/introduction-to-graph-databases/>
- Dietrich, J., Jones, N. and Wright, J., 2008, "Using social networking and semantic web technology in software engineering – Use cases, patterns and a case study", *Massey University Institute of Information Sciences and Technology, Palmerson North, New Zeland, January 2008*.
- Guia, J., Soares, V. and Bernardino, J., 2017, "Graph Databases: Neo4J Analysis", *19th International Conference on Enterprise Information Systems*.
- InfiniteGraph interface, <http://image.slidesharecdn.com/igoverviewdbtaaug2011-110830134434-phpapp02/95/webinar-an-introduction-to-infinitegraph-and-connecting-the-dots-in-big-data-17-728.jpg?cb=1314711996>
- InfiniteGraph, <http://www.objectivity.com/products/infinitegraph/>
- Mehra, A., 2017, "Understanding the CAP Theorem", <https://dzone.com/articles/understanding-the-cap-theorem>
- Neo4J, "Why Graph Databases?", <https://neo4j.com/why-graph-databases/>, (Accessed: 2 of March of 2018)
- Neo4J, www.neo4j.com
- OrientDB, <https://orientdb.com/>
- Ranking of Graph DBMS, 2018, db-engines, <https://db-engines.com/en/ranking/graph+dbms>
- Robinson, I., Webber, J. and Eifrem, E., 2015, *Graph Databases, 2nd Edition, O'Reilly Media Inc., California*
- Top Graph Databases, 2017, Predictive Analytics Today, <https://www.predictiveanalyticstoday.com/top-graph-databases/>
- Wu, M., 2017, "What Are the Major Advantages of Using a Graph Database?", <https://dzone.com/articles/what-are-the-pros-and-cons-of-using-a-graph-databa>