Linking Graph Databases and Semantic Web for Reasoning in Library Domains

Davide **Di Pierro**[1], Domenico **Redavid**[2] and Stefano **Ferilli**[3]

Abstract

The need for digital libraries to store, catalogue, and manipulate documents is growing and the requirements are increasing daily. Much research is underway to discover effective ways to store and keep track of documents, their properties and uses. In this context, we propose the use of Labeled Property Graphs (LPGs) to store digital archives. The use of these graphs allows us high efficiency since the graphs are navigable very quickly thanks to Database Management Systems (DBMSs) (such as Neo4j) especially developed to guarantee high performance. On the other hand, this technology is not natively suited to infer new knowledge automatically. In this paper, we propose a method to use Semantic Web tools to enrich our knowledge base. This way, new operations are available, not possible with only graph databases. The proposed approach has been integrated into the GraphBRAIN (GB) system, a tool able to manage ontologies.

Keywords

digital libraries, graph databases, semantic web, automated reasoning

Linking Graph Databases and Semantic Web for Reasoning in Library Domains

1. Introduction

The Semantic Web (SW) is inspired by a vision of the current Web which has been in the background since its inception, and which is influenced by earlier work dating back to Vannevar Bushs idea of the memex machine in the 1940s (based on a universal library, complete with a searchable catalogue) [1]. Tim Berners-Lee originally envisioned the World Wide Web as including richer descriptions of documents and links between them [2]. However, in the effort to provide a simple, usable and robust working system, which could be used by everyone out-of-the-box, these ideas were put to one side, and the simpler, more human-mediated Web which we know today resulted. The bigger vision found expression in an article written by Tim Berners-Lee in Scientific American in May 2001 [3]. In this article, they provide a compelling vision of a world where instead of people laboriously trawling through information on the Web and negotiating with each other directly to carry out routine tasks such as scheduling appointments, finding documents and locating

services, the Web itself can do the hard work for them. For thousands of years, libraries have allowed humanity to collect and organize data and information, and to support the discovery and communication of knowledge, across time and space. Coming together in this Internet Age, the worlds societies have extended this process to span from the personal to the global, as the concepts, practices, systems, and services related to Library and Information Science unfold through digital libraries. Scientists, scholars, teachers, learners, and practitioners of all kinds benefit from the distributed and collaborative knowledge environments that are at the heart of the digital library movement. National Science Foundation defined digital libraries as instruments to basically store materials in electronic format and manipulate large collections of those materials effectively. Research into digital libraries is research into network information systems, concentrating on how to develop the necessary infrastructure to effectively mass-manipulate the information on the Net. Research in digital libraries has become an academic discipline, with computer scientists working alongside economists, sociologists, lawyers, and librarians. An interdisciplinary body of expertise emerged [4]. The availability of data makes the use of SW technologies attractive. Several works tried to integrate SW with graph databases, a powerful means recently introduced in DL infrastructure, but we consider that none of them is satisfactory for our purposes. The difficulty in integration lies in the diversity of models used. An LPG graph consists of nodes and arcs to which attributes can be associated, while a Resource Description Format (RDF) graph consists of statements, e.g., subject-predicate-object triples. Different types of mapping are needed, depending on whether we have RDF information to import into LPG or vice versa.

From a more technical point of view, the design of software applications in general and for digital libraries, in particular, are strongly influenced by the persistence model used. The availability of reliable graph databases, therefore, opens new perspectives in the realization of applications. Graph databases allow importing different sources of data in a very flexible way due to the non-static nature of the data schemes. Graph databases were not created as a technology for the SW. In particular, the SW is based on the RDF model different from the Property Graph model of graph databases.

In this paper, we propose an approach for knowledge enrichment in a graph database. To do so, it will be necessary to map the LPG data in RDF and make use of Ontology Web Language (OWL) ontologies.

In the next sections, we will see in detail how we realized the proposed approach inside the GraphBRAIN tool and which functionalities can be realized through the use of an SW reasoner.

This paper is organized as follows: Section 2 treats the state-of-art of. Section 3 describes the building blocks of our proposal, particularly focusing on the proposed ontology for Library domains, the translation mechanisms and the reasoning. Section 4 summarizes and concludes the proposal.

2. Related Works

Even if not strictly applied to digital libraries, in the literature, we found several attempts to integrate, and, possibly, extract new knowledge or create a mapping between graph databases and the SW. Many of them, however, did not consider the idea of putting them together for exploiting the advantages of both.

neosemantics [1] is a Neo4J plugin that enables the use of RDF and its associated vocabularies like (OWL, RDFS, SKOS and others) in Neo4j. At the current stage neosemantics offers an import/export mapping starting from an RDF graph, but starting from an LPG graph the mapping of all contained information is not guaranteed. On the other side, native RDF repositories, like GraphDB [2], are conceived to store RDF model rather than LPG. We aim to exploit the characteristics of the property graph by using its available functionalities and enabling SW tools to enrich the knowledge in the LPG. In [5], it was proposed an approach that consists of directly mapping RDF databases into PG databases. They demonstrate empirically and formally, that the mappings had an efficient implementation to process large datasets. The main advantage of this proposal was the information preservation, that is, there existed inverse mappings that allowed recovering the original databases without losing information. Currently, this is not guaranteed in our system because we focused on exposing the data to apply reasoning. There are several proposals, like [6], that performed a mapping between the two formalisms based directly on the queries that could be performed on one type of graph or another. For instance, a mechanism to execute SPARQL queries directly on graph database (db) was described. Currently, our system is only able to expose data in RDF, but we have planned to enhance it to be queried with SPARQL. Hartig [7] proposed two transformations between RDF* and property graphs. RDF* is a syntactic extension of RDF which is based on reification. The first transformation mapped any RDF triple as an edge in the resulting property graph. Each node had the kind attribute to describe the type of a node (e.g. IRI). The second transformation distinguished data and object properties. The former was transformed into node properties and latter into edges of a property graph. The limitation of the second transformation was that metadata triples could not be transformed. The shortcoming of this approach is that RDF* is not supported by the majority of RDF triplestores and requires conversion of existing RDF data beforehand. [8] Even if not correlated with the mapping between graph databases and SW tools, in [9] there is an attempt to translate data used in the context of digital libraries into RDF. In [10], there is a proposal to combine the benefits of the two models creating an intermediate abstraction layer called Singleton Property Graph (SPG). The SPG layer sit on top of the RDF and simulated the property graph model. This middle layer was able to interpret SPARQL queries. In this work, however, the problems related to control over the information that you could store in the graph db and the management of them were not analyzed.

---

[1]Neo4j neosemantics library, https://neo4j.com/labs/neosemantics
[2]OntoText Graphdb, https://graphdb.ontotext.com/documentation
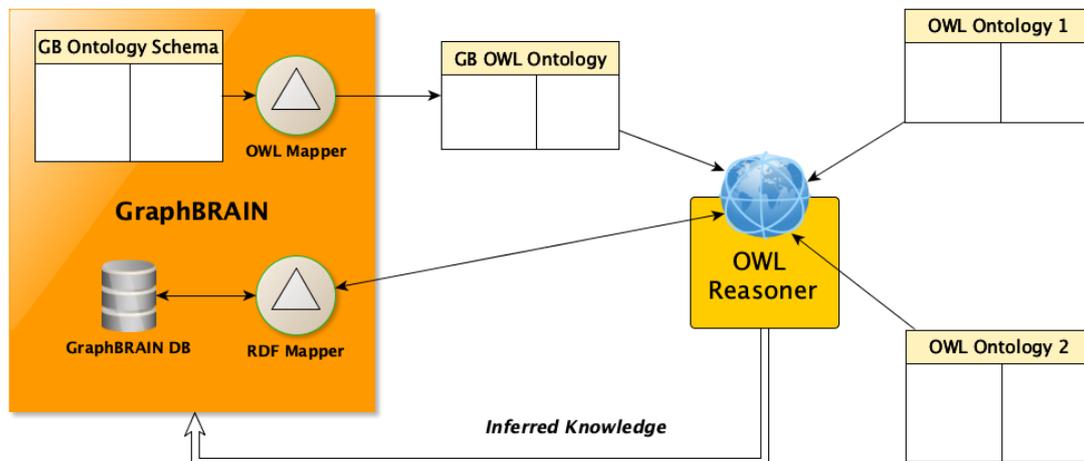
3. Building Blocks

Figure 1: Building blocks

For the first implementation of our proposal, we leveraged several previous works and systems from our past research. The entire architecture we will describe has not yet been fully implemented, in this work we propose a general solution. Our initial prototype was integrating and combining them for carrying out the many activities required. Knowledge storage and management was carried out using GraphBRAIN [11][3]. It is a general-purpose system aimed at supporting all stages and tasks in the lifecycle of a knowledge base from knowledge base design, to knowledge acquisition, to knowledge organization and management, to (personalised) knowledge fruition and delivery. To ingest the document and process them to fill the knowledge graph, a (semi)automatic system was required. Much help for the layout and logical description may come from research on Document Image Analysis, while for content and conceptual description several methods and tools are available from the Natural Language Processing community. We are currently using the DoMInUS system [12] that includes solutions from both fields, for preliminary processing, plus an interactive interface to adjust the automatic results and fine-tune what is to be reported in the formal description of the document. We briefly remind our methodology and what are we going to discuss here. We want to enrich our graph database with information that can be inferred with SW techniques. For this reason, we need to work at two different levels of abstraction: 1) render LPG data in RDF and 2) describe their semantics in an OWL ontology. In this way we can reach our goals: exploit the efficiency of the graph database and enable SW technologies, in particular the SW reasoning. For completing the integration, we first need an

---

[3]available at: http://193.204.187.73:8088/GraphBRAIN/

ontology that describes our concepts. We have Digital Library data as nodes and arcs in the graph database. We also have GraphBRAIN schemes used to support these data. These schemes represent general ontology definitions for the LPG and they are stored in an eXtensible Markup Language (XML) format whose Data Type Definition (DTD) has been described in [13]. In the GraphBRAIN system, there is the possibility to generate, from of our ontology schemas, a minimal OWL ontology. Finally, we also designed a methodology for translating actual data taken from LPG into RDF triples. Obtained data in RDF and concept and relation in OWL an SW reasoner can be used to check the consistency of the knowledge base and infer new knowledge. The building blocks of this proposal are shown in Figure 1. During the description of the various building blocks, we will carry out an example and follow all the steps to demonstrate one of the possible operations that can be carried out using this architecture.

3.1. GraphBRAIN

Since ontology and knowledge graph management is the core of our proposal, we will describe in some more detail the GraphBRAIN component. It provides a tool to design and collaboratively populate knowledge graphs, and advanced solutions for their fruition, consultation and analysis.
A distinguishing feature of GraphBRAIN is its mixing Knowledge Representation and Graph Database technologies to take advantage of both, the efficiency (scalability, storage optimization, etc.) of the latter and the flexibility and power of the former for storing and handling knowledge. The graph DBMS implementing the knowledge base is Neo4j[4]. While it is schema-free (the user may apply any label and/or attribute to any single node or arc), GraphBRAIN imposes the use of a predefined XML data schema (GB schema). It represents a form of ontology schema, so that only data that are compliant with it may be added to the graph, and all functions are driven by it. The GB schema defines what (kinds of) knowledge can be represented in the knowledge base, how it is to be described, and how it can be exploited. While maintaining a single, overall graph to store the data, GraphBRAIN is designed to allow the use of different GB schemes on it, each expressing a different domain or perspective. So, using a specific GB scheme one has only a partial view of the knowledge. However, different GB schemes may share the same classes (uniquely determined by name, so that alignment is easily obtained), possibly defining different sets of attributes for them. E.g., classes such as Person, Place, or Document are expected to appear in different domains/schemes; in one schema an attribute of Person might be its professional title, which is not needed in others. Individuals of shared classes act as bridges, allowing the computations in a domain to reach and exploit information coming from other domains. This implies that adding new items under one domain will automatically enrich also the other domains whose instances can be reached via some path in the graph. To the best of our knowledge, this is a completely

---

[4]Neo4j Graph Data Platform - www.neo4j.com

innovative setting, that enables cross-fertilization among, and knowledge reuse across, different domains. It is fundamental to enforce our 'contextual' perspective. Furthermore, GraphBRAIN can import/export its ontologies and knowledge graphs from/to Semantic Web languages, e.g., OWL and RDF.

In general, the functions provided by GraphBRAIN bring to cooperation many different Artificial Intelligence tasks, techniques and approaches for improving knowledge management and (personalized) fruition by users, including database technology, data mining, machine learning, natural language processing, recommendation, collaborative and social interaction tools, social network analysis. This allows it to find relevant, personalized and non-trivial information. E.g., a social approach can be used to build and integrate GB ontology schemes and user models can be used to guide data mining.

This is useful for some functionalities based on the statistic field outside of the SW technologies application but is required and useful. In particular, the Link Prediction algorithms works on pieces of graph allowing to display the nodes and arcs. Over these nodes and arcs the following algorithms can be executed: Adamic-Adar Index, Common Neighbor, Resource Allocation Index and Katz.

3.2. GraphBRAIN schema

Here we show the proposed GB schema. It is not strictly related to SW ontology but is LPG oriented since it contains some restrictions needed to the application level. Furthermore, GB scheme could be extended to represent semantic information with languages different from those offered by SW, like Inductive Logic Programming or datalog. For the GB schema definitions, we refer to our previous work, which will certainly be extended and deepened [14]. From the semantic abstraction layer, we refer to a GB schema and a related use case within GraphBRAIN. Here we will give an overview of its artefact based on the meaning rather than a formal representation, trying to explain the underline semantics. It includes several general concepts relevant to the library domain, among which **User** (whose identity may be unknown), **Person** (whose identity is known, possibly associated to a user), **Author** (who is a person who writes something) and **Place** (a place in which something happens). These concepts can be considered general, widely used concepts and reusable in most domains. Then, there are the domain-specific concepts for the library. Among the most obvious, **Document** (to describe a piece of knowledge), **Book** (a particular case of Document), **Stream** (stream of data), **Multimedia** (e.g., images, audios and videos), **Award** (for prizes and recognitions) and many other minors. The GB ontology schema must be able to describe the contents of sources since they are the main elements for which this ontology is meant to be used. Documents may take many very different forms, requiring a more elaborate structure for this concept. The most prominent kinds of documents are books, papers, articles, blogs, websites, posts and many other possible sub-concepts at different levels. Multimedia objects should be distinguished as well (**Music**, **Image**, **Audio** and **Video**). For fine-grained handling of documents, we also provides concepts to describe their layout (**Page**, **Table**, ...)

```
71⊖          <entity name="Document">
72⊖              <attributes>
73                   <attribute name="name" mandatory="true" datatype="string"/>
74                   <attribute name="isbn" mandatory="false" datatype="string"/>
75                   <attribute name="firstAuthor" mandatory="false" datatype="string"/>
76                   <attribute name="edition" mandatory="false" datatype="string"/>
77                   <attribute name="copyright" mandatory="false" datatype="date"/>
78                   <attribute name="format" mandatory="false" datatype="string"/>
79                   <attribute name="length" mandatory="false" datatype="string"/>
80                   <attribute name="date" mandatory="false" distinguishing="true" datatype="date"/>
81                   <attribute name="originalPrice" mandatory="false" datatype="real"/>
82              </attributes>
```

Figure 2: The Document concept in GB schema

and logical (**Title**, **Section**, ...) structure, their text and its grammatical structure (**Sentence**, **Subject**, **Object**, ...), and the concepts (**Category**) and Named Entities (**Person**, **Place**, etc.) they mention. Some other concepts can be used for auxiliary purposes like **Artifact** (including artisanship and industrial items, or works of art, such as statues and paintings), **Device** (including simple tools, such as hammers, and more complex systems, such as computers), **Software** (again, with a taxonomy of sub-concepts for the different kinds of software) and **IntellectualWork** (including tools, algorithms, programming languages, technologies, theoretical models and so on). We provide many relationships to connect instances of these concepts. Some are needed to organize documents into layout and logical components (e.g., **has** links a **Document** to its layout and logical components; **partOf** structures components into sub-components, etc.) and to associate their content with generic concepts or specific named entities (mentions). **LayoutComponent**s also have a spatial organization, expressed by relationships **leftOf** and **above**. Authors and documents are connected to them through the **developed** relationship. Other relationships can be easily figured out by readers (e.g., those linking **Person**s to **Organization**s, or those describing the syntactic and semantic structure of the text.

As an example, we take the Document concept having the following attributes among the others (see Figure 2): name, ISBN, firstAuthor. There are a lot of other attributes but we refer to these three just for the example we mean to explain.

Table 1
Translations from XML to OWL

Entity	owl:Class
Entity Attribute	owl:DatatypeProperty
Relationship	owl:ObjectProperty
Subject of Relationship	owl:ObjectProperty Domain
Object of Relationship	owl:ObjectPropertyRange

3.3. From GB schema to OWL Ontology

To apply the SW reasoning, as the first step we need to generate an OWL ontology from GB schema. With this aim, we have designed a mapping between their entities reported in Table 1. The main step to bring concepts and relationships into the SW is to assign them a unique URI. For our scopes, we use the following namespace: "https://gbnamespace#" whose given abbreviation is dl. Hence, every concept and relationship described in the previous section will contain that namespace when translated into OWL definitions. GraphBRAIN is also able to import OWL ontologies. In this case, the mapping in the Table 1 is applied from OWL to XML. This procedure is limited to the entities that are representable in GB schema. The OWL classes and datatype properties of the example are, for a matter of size, shown in part below.

```xml
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY dl "http://www.graphbrain.it#" >
        <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>

<rdf:RDF xmlns="http://www.graphbrain.it#"
    xml:base="http://www.graphbrain.it"
    xmlns:dl="http://www.graphbrain.it#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://www.graphbrain.it"/>
        <owl:Class rdf:about="&dl;Document"/>
    <owl:DatatypeProperty rdf:about="&dl;firstAuthor">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
        <rdfs:domain>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&owl;topObjectProperty"/>
                <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1
                </owl:qualifiedCardinality>
                <owl:onClass rdf:resource="&dl;Document"/>
            </owl:Restriction>
        </rdfs:domain>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

    ...

</rdf:RDF>
```

3.4. From LPG to RDF

Complementary to the ontology in OWL, we need data in the form of RDF triples, with the subject-predicate-object structure. There are several techniques to achieve this. We decided to exploit the potential of Neo4j that allows us, through a Cypher query, to extract data from a db in JavaSript Object Notation (JSON) format using

the Neo4j APOC library[5]. In this way, we can parse the JSON file and render all the extracted nodes and arcs (with their respective attributes) into RDF. To do this, we follow the mapping rules listed in Table 2. We can render in RDF the whole LPG or pieces of a graph. By extracting a part of the graph related to what has been required to render, we can prepare to respond more quickly to the next requests that presumably will be made from the requester (for example from an SW reasoner). We can take new instances from LPG with different techniques. For example, all nodes and arcs far k hops from a central node.

We do not keep a copy of the database as RDF triples, RDF triples are generated on the fly from a request by a requester. Referring to our example, suppose we obtain from the graph the following node rendered in RDF where the information about the first author is missing:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22−rdf−syntax−ns"
    xmlns:dl="https://gbnamespace\#">


<rdf:Description
    rdf:about="https://gbnamespace/DivineComedy">
        <dl:id>6056</dl:id>
        <dl:isbn>978−81−7525−766−5</dl:isbn>
        <dl:name>Divine Comedy</dl:name>
    </rdf:Description>
</rdf:RDF>
```

3.5. OWL Reasoner applications

After extracting some nodes and reading the ontology, one of the available SW reasoners can be used. The operations possible with SW reasoners mainly concern hierarchies among classes or instance checking. The most useful operation can be

---

[5]Export Neo4J graph to JSON, https://neo4j.com/labs/apoc/4.1/export/json

Table 2
Translations from LPG to RDF

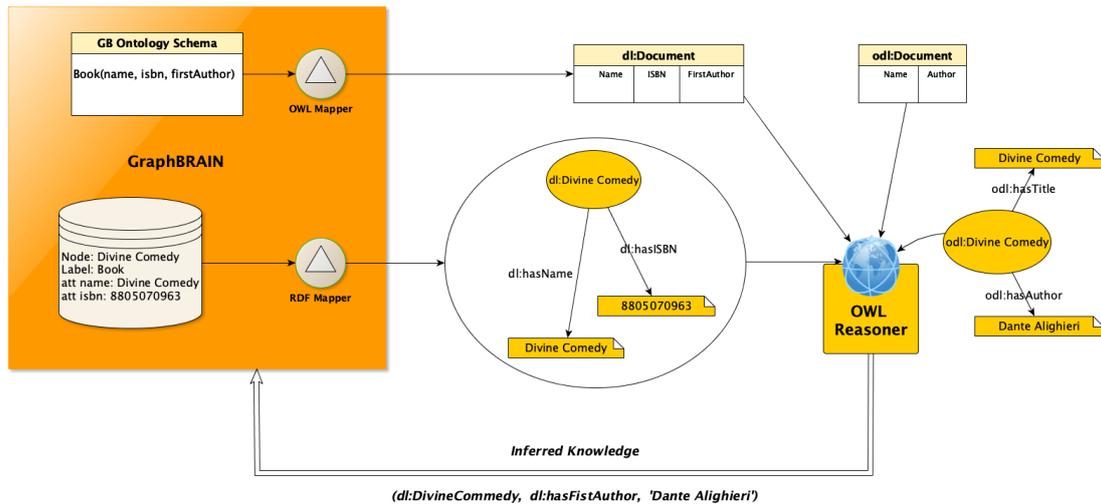| LPG Element	RDF Element
Node	rdf:subject
Arc	rdf:predicate
Attribute on node	rdf:predicate between the rdf:subject (node) and the literal (Attribute value)
Attribute on relationship	rdf:predicate between the rdf:statement representing the triple (relationship) and the literal (Attribute value)

Figure 3: Example of how the information is filled within nodes

to check the consistency of the knowledge base during the input of data in the LPG using GraphBRAIN.

Furthermore, having the possibility of reading other OWL ontologies and related RDF triples, we can merge information not contained in our graph database.

In this first proposal, we map the external OWL ontology classes with GB ontology schema applying a (sub)string matching and imposed owl:samAs properties when a match is found. In this way, LPG instances will be rendered as owl:individuals or the mapped OWL ontology. Remind that this is just a first proposal of the architecture and, since we have not developed the system in its totality, we can further investigate advanced techniques for this relevant aspect of this architecture. Turning back to our example, suppose to import (among the many sources) an OWL ontology (whose abbreviated namespace is "odl") that states in its RDF store that the Author of Divine Comedy is "Dante Alighieri". The alignment procedure has found a correspondence between the attributes odl:Author and dl:FirstAuthor. Since FirstAuthor is not a mandatory field, we have no author given for the Book "Divine Comedy" in the LPG. Applying the reasoner, this information is suggested to the GraphBRAIN administrator that can evaluate if adding it to the LPG. This is just a trivial example, reported completely in Figure 3 but it is easy to imagine, in the same context, if we have two different information coming from different sources. These types of situations are very common when handling digital libraries where large communities can put data and metadata in the graph with limited control from GraphBRAIN. The extracted information is not stored anywhere if the administrator decides to not insert them in the database. In this way, we are sure to avoid unwanted or unpredictable changes to our source.

4. Conclusions

In this paper, we have proposed a system that can effectively combine graph databases and the SW to deduct new knowledge which is something not possible only with Neo4j. We proposed to manage an LPG in Digital Library Domain with GraphBRAIN to have some level of control on the contained information. An approach implemented in GraphBRAIN for the generation/import of OWL ontologies and a mapping between LPG data and RDF formalism is given. Many improvements can be applied and we are working to integrate them. Indeed, this work lends itself well to numerous extensions. First, there are several ways through which significant nodes of a graph can be extracted. In the GraphBRAIN system, some algorithms are already available for visualizing parts of the graph, like Katz Centrality and PageRank Centrality. There is a further possibility of using the SW reasoner to infer new knowledge.

References

[1] B. Matthews, D. Brickley, L. Dodds, Semantic web technologies world wide web consortium, ResearchGate (2005).

[2] T. B. Lee, Information management: A proposal, 1989. URL: http://www.w3.org/History/1989/proposal.html.

[3] T. B. Lee, J. Hendler, O. Lassila, The semantic web, Scientific American, a division of Nature America, Inc. 284 (2001) 34–43.

[4] W. Y. Arms, Digital Libraries, MIT Press, 2001.

[5] R. Angles, H. Thakkar, D. Tomaszuk, Mapping rdf databases to property graph databases, IEEE Access 8 (2020) 86091–86110.

[6] H. Thakkar, D. Punjani, J. Lehmann, S. Auer, Two for one: Querying property graph databases using sparql via gremlinator, in: Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems (GRADES) and Network Data Analytics (NDA), Association for Computing Machinery, 2018.

[7] O. Hartig, Reconciliation of rdf* and property graphs, 2014.

[8] R. Angles, H. Thakkar, D. Tomaszuk, Rdf and property graphs interoperability: Status and issues, in: AMW, 2019.

[9] C. Meghini, N. Spyratos, T. Sugibuchi, J. Yang, A model for digital libraries and its translation to rdf, Springer 3 (2012) 51–59.

[10] V. P. Nguyen, H. Y. Yip, H. Thakkar, Q. Li, E. Bodenreider, B. Bodenreider, O. Bodenreider, Singleton property graph: Adding a semantic web abstraction layer to graph databases, 2019.

[11] S. Ferilli, D. Redavid, The GraphBRAIN System for Knowledge Graph Management and Advanced Fruition, volume 12117, 1st ed., Springer, 2020.

[12] S. Ferilli, F. Esposito, T. M. A. Basile, D. Redavid, I. Villani, DOMINUS plus - document management intelligent universal system (plus), in: M. Agosti,

F. Esposito, C. Meghini, N. Orio (Eds.), Digital Libraries and Archives - 7th Italian Research Conference, IRCDL 2011, Pisa, Italy, January 20-21, 2011. Revised Papers, volume 249 of *Communications in Computer and Information Science*, Springer, 2011, pp. 123–126.

[13] S. Ferilli, Integration strategy and tool between formal ontology and graph database technology, Electronics 10 (2021).

[14] S. Ferilli, D. Redavid, An ontology and knowledge graph infrastructure for digital library knowledge representation, in: Digital Libraries: The Era of Big Data and Data Science, volume 1177 of *Communications in Computer and Information Science (CCIS)*, Springer International Publishing, 2020, pp. 47–61.