RDF and Property Graphs Interoperability: Status and Issues

Renzo Angles[1,2], Harsh Thakkar[3], Dominik Tomaszuk[4]

[1] Universidad de Talca, Chile
[2] Millennium Institute for Foundational Research on Data, Chile
[3] University of Bonn, Germany
[4] University of Bialystok, Poland
rangles@utalca.cl[1], thakkar@cs.uni-bonn.de[3], d.tomaszuk@uwb.edu.pl[4]

Abstract. RDF and Property Graph databases are two approaches for data management that are based on modeling, storing and querying graph-like data. In this paper, we present a short study about the interoperability between these approaches. We review the current solutions to the problem, identify their features, and discuss the inherent issues.

1 Introduction

RDF [24] and graph databases [37] are two approaches for data management that are based on modeling, storing and querying graph-like data. Several database systems based on these models are gaining relevance in the industry due to their use in several domains where graphs and network analytics are required [6].

Both, RDF and graph database systems are tightly connected as they are based on graph-oriented database models. On the one hand, RDF database systems (or triplestores) are based on the RDF data model [24], their standard query language is SPARQL [19], and there are languages to describe structure, restrictions and semantics on RDF data (e.g. RDF Schema [13], OWL [18], SHACL [25], and ShEx [11]). On the other hand, most graph database systems are based on the Property Graph (PG) data model [7], there is no standard query language (although there are several proposals [4]), and the notions of graph schema and integrity constraints are limited [32]. Therefore, these two groups of systems (in particular the latter) are dissimilar in data model, schema, query language, meaning and content.

Given the heterogeneity between RDF and graph database systems, it results necessary to study the interoperability among them, i.e. the ability of these systems to exchange data, information (structure and semantics) and knowledge (constraints and business rules).

The main objective of this paper is to present an overview of the research concerning the interoperability between RDF and property graph databases. First, we clarify the notion of database interoperability, identifying three types: syntactic interoperability, semantic interoperability, and query interoperability (Section 2). Second, we present a short review of the current approaches and works, including data format transformations, data and/or schema exchange, and query translations. (Section 3). Third, we isolate and discuss the main issues and challenges in the topic (Section 4).

2 Database Interoperability

The term "Interoperability" was introduced in the area of information systems, and it could be defined as the the ability of two or more systems or components to exchange information, and to use the information that has been exchanged [3]. In the context of data management, interoperability is concerned with the support of applications which exchange and share information across the boundaries of existing databases [38].

Providing interoperability between database models, systems and applications is a very concrete and pragmatic problem, which stems from the need of reusing existing systems and programs for building new applications [38]. Data and information interoperability is relevant for several reasons, including:

- Promotes data exchange and data integration [30];
- Allows to have a common understanding of the meanings of the data [22];
- Allows the creation of information and knowledge, and their subsequent reuse and sharing [40];
- Facilitates the access to a large number of independently created and managed information sources of broad variety [40];
- Facilitates the reuse of available systems and tools [38];
- Allows to explore the best features of different approaches and systems [31];
- Enables a fair comparison of database systems by using benchmarks [5];
- Supports the success of emergent systems and technologies [38];
- It is a crucial factor for the development of new information systems [29].

One can define several forms of interoperability in information systems [28]. For instance, focusing in the dimension of heterogeneity, Sheth [40] defined four levels of interoperability: system, syntax, structure and semantic. The system level concerns the heterogeneity of computer systems and communications. The syntax level considers machine-readable aspects of data representation (i.e. data formats and serializations). The structure level involves data modeling constructs and schematic heterogeneity. The semantic level requires that the information system understand the semantics of the use's information request and those of information sources.

In the context of Web Languages and Ontologies, the syntactic interoperability means that the applications can take advantage of parsers and APIs providing syntactical manipulation facilities. Additionally, semantic interoperability implies that applications can understand the meaning of representations and thus can setup automatically mappings between different representations by content analysis [33].

In the context of databases, interoperability can be divided into syntactic, semantic and query interoperability. *Syntactic interoperability* refers to the ability of a database system to use data from other database system [23]. It could means that both database systems are able to exchange information, although they may not being aware of the meaning of such information. *Semantic interoperability* can be defined as the ability of database systems to exchange data in a meaningful way. It implies that the systems have a common understanding of the meanings of the data [22]. *Query interoperability* implies the existence of methods to transform different query languages or data accessing methods between two systems. It means that a query in the source database system can be translated into one that can be directly executed on the target system [48].

3 RDF and Property Graphs interoperability

In general terms, syntactic interoperability between RDF and PG databases means data exchange at the level of serialization formats, semantic interoperability implies the definition of data and schema mappings, and query interoperability implies query translations among SPARQL and property graph query languages. This section presents a review of the approaches and methods proposed for these types of interoperability.

3.1 RDF databases

Every RDF database system is designed to support the Resource Description Framework [24], a W3C standard created to describe web resources, although it could be used in any application domain. RDF defines a data model which is based on the notion of RDF triple. An RDF triple is a tuple formed by a subject, a predicate, and an object. The *subject* denotes a resource, the *predicate* refers to an attribute or relationship of the subject, and the *object* defines the value for such property. A collection of RDF triples could be visualized a graph where the subjects and objects are represented as nodes, and the predicates are represented as edges. An RDF database could be considered as a collection of RDF graphs. RDF reification is a feature which means to create triples about triples, here metadata (e.g. temporal, uncertainty and trust metrics).

The RDF Schema vocabulary [13] provides a simple way to describe the structure of an RDF database. In this case, the schema is described as a collection of a resource classes and property classes. Moreover, the classes could be hierarchically organized by using *subclass* and *subproperty* relationships. More complex restrictions can be expressed in languages like OWL [18], SHACL [25] and ShEx [34]. These languages provide semantic interpretations that allow to infer additional triples. This feature is called RDF(S) entailment.

SPARQL is the standard query language to retrieve and manipulate RDF data. The first version (SPARQL 1.0 [35]) provides basic operators to express graph pattern matching. The second version (SPARQL 1.1 [19]) adds features like explicit negation, path expressions, subqueries and aggregate operators.

3.2 Property graph databases

Most of the current graph database systems have been designed to support the property graph data model. A property graph [7] is a directed labelled multi-graph with the special characteristic that each node or edge could maintain a set (possibly empty) of properties. A property is formed by a name and a value.

The notion of schema for a property graph database has not been developed in practice, although some systems use the notions of node types and edge types. Integrity constraints are also in development. In [32], the authors mention three types of integrity constraints: inherent constraints, explicit constraint and implicit constraint. Additionally, we found node/edge/property constraints, path constraints, and graph pattern constraints with property values [12].

In spite of the extensive research on querying graph databases [4], there is no standard query language for property graphs. A recent publication, called the GQL manifest [2], proposes to define and standardize one property graph query language by fuzing the best of three query languages: Cypher [1], PGQL [36] and G-CORE [8].

3.3 RDF-PG Syntactic interoperability

Assume that the syntactic interoperability is given by the facilities to transform data from one format to another. Hence, the main requirement to support syntactic interoperability is the existence of data formats (i.e. a syntax for encoding data stored in a database), over which transformation methods can be implemented.

Turtle, TriG, RDF/XML, RDF/JSON and JSON-LD are data formats for encoding RDF data. In contrast, there is no data format to encode property graphs. Given this restriction, some systems use graph data formats (like GraphML, DotML, GEXF, GraphSON), but none of them is able to cover all the features presented by the property graph data model. YARS-PG [47] is a recent proposal of data format which was designed to be simple, extensible and platform independent, and to support all the features provided by the current database systems based on the property graph data model.

Given a source data format S and a target data format T, the first option to support syntactic interoperability is to define a textual mapping from S to T. Note that the schema of the database is not considered in the transformation. Hence, the structure, semantics and restrictions of the source data could not be preserved by the translated data.

Hartig [20] proposes two transformations between RDF* and property graphs. RDF* is a syntactic extension of RDF which is based on reification. The first transformation maps any RDF triple as an edge in the resulting property graph. Each node has the "kind" attribute to describe the type of a node (e.g. IRI). The second transformation distinguishes data and object properties. The former are transformed into node properties, and latter into edges of a property graph. The limitation of the second transformations is that metadata triples cannot be transformed. The shortcoming of this approach is that RDF* isn't supported by majority of RDF triplestores (except Blazegraph and the most recent addition, AnzoGraph) and requires conversion of existing RDF data beforehand.

Schätzle et al. [39] propose a mapping which is native to GraphX (a parallel processing system implemented on Apache Spark). The proposed graph model is an extension of the regular graph, but lacking the concept of attributes. The mapping uses an special attribute *label* to store the node and edge identifiers, i.e. each triple $t =$ *(s, p, o)* is represented using two vertices v_s, v_0, an edge (v_s, v_o) and properties $v_s.label = s$, $v_o.label = o$, $(v_s, v_o).label = p$. The proposed method does not address blank nodes or RDF entailment.

Nyugen et al. [27] proposed *LDM-3N* (labeled directed multigraph-three nodes), a graph model for RDF data. It is an extension of the regular graph, without the concept of attributes, and represents each triple element as separate nodes, thus three nodes (3N) . The *LDM-3N* graph model is used to address the Singleton Property (SP) based reified RDF data.

Tomaszuk [46] presented an approach that uses the *YARS* serialization for transforming RDF data into property graphs. This approach basically performs a transformation between encoding schemes and does not consider the RDF schema and its entailments. This approach has several implementations, eg. neo4j-yars[5] and TTL2YARS[6].

With respect to the methods to transform property graphs into RDF graphs, the literature is very restricted. The current methods [16,20] are based on reification. In the simplest case, for each edge in the property graph there will be a blank node (in the RDF graph) containing at least three nodes (resources or literals) and three edges (properties). Such elements will be necessary to describe all the information of the original edge.

A additional approach to provide syntactic interoperability is the use of an intermediate data format. It is possible to find some tools to transform RDF into other formats and vice versa, eg. Triplify [9] for relational data, GRDDL [14] for XML and CSVW [42] for tabular data. However, to the best of our knowledge, there is not study about the subsequent transformation to property graphs.

3.4 RDF-PG Semantic interoperability

Semantic interoperability between databases means that both, source and target systems, are able to understand the meaning of the data to be exchanged. It implies that both, data and schema must participate of the transformations method.

A common approach to support semantic interoperability is the definition of data and schema transformation methods. The schema transformation method takes as input the schema of the source database, and generates a schema for the target database. Similarly, the data transformation method allows to move the data from the source database to the target database, but taking care of the target schema. The transformation methods can be implemented by using data

[5] `https://github.com/lszeremeta/neo4j-sparql-extension-yars`
[6] `https://github.com/lszeremeta/ttl-to-yars`

formats or data definition languages. To the best of our knowledge, there is no method that support data and schema transformations between RDF and PGs.

A additional approach is the use of a data transformation language. For instance, XSPARQL [10], and SPARQL Template Transformation Language (STTL) [15] are languages that allow data transformation between RDF and other languages or formats. In the opposite direction, RML [17] is a generic language which allows to define mappings from heterogeneous sources to RDF. In a recent article [26], the authors present the Graph to Graph Mapping Language (G2GML) for mapping RDF graphs to property graphs. This language can be processed by an implementation called G2G Mapper (available on `https://github.com/g2gml`). There is no formal definition nor analysis of the features of this transformation language.

3.5 RDF-PG Query interoperability

Query interoperability between RDF and property graph databases is a current issue due to the lack of a standard query language for property graphs. Gremlinator [44,45] is a tool that translates SPARQL queries into Gremlin pattern matching traversals. Gremlin is a popular language used by some graph database systems and graph processing frameworks. Gremlinator [44] has been successfully integrated as a plugin of the famous Apache TinkerPop graph computing framework[7]. Given the above, the openCypher initiative is working on the Cypher to Gremlin translation (`https://github.com/opencypher/cypher-for-gremlin`).

Hartig et al. [21] defined extensions of the SPARQL query language that capture an alternative approach to represent statement-level metadata that can be used in property graphs (see [20]). This proposal, called SPARQL* is an RDF*-aware extension that introduces new features that enable users to directly access metadata triples in queries.

4 Issues and challenges

Based on our literature review about RDF and property graphs interoperability, we identified several issues and challenges which will be discussed in this section.

4.1 Syntactic interoperability

– There is no standard data format for encoding property graphs. This is a crucial issue to support syntactic interoperability.
– The most RDF serializations are triple-centric, while the most PG serializations represent graph as lists of nodes and edges.
– Despite the serializations based on JSON or XML in both models, the syntaxes used are difficult to map.

[7] The `sparql-gremlin` plugin of the Apache TinkerPop framework available on Github – (`https://github.com/apache/tinkerpop/tree/master/sparql-gremlin`)

– The support for multi-values is different in the models. A property graph just support arrays, while RDF provides different types of lists.
– The RDF data model allows metadata about properties, i.e. edges between edges are allowed. Although this feature is not common in real data, a data mapping should be able to manage it. Note that a property graph does not support multi-level metadata.
– RDF reification leads to an explosion in the size of the resulting graph. This can be avoided by implementing a "smart" transformation that is able to recognize a set of triples describing a reification, and map them to a single node in the property graph.

4.2 Semantic interoperability

– The RDF model presents features with special meaning (or semantics) that cannot be modeled by the property graph data model (at least not in a easy way). Blank nodes, reification, and entailment are some of these features.
– Usually, an RDF database contains a mix of data and schema. In such case, it is necessary to decide whether to extract the schema (and transforming it independently), or processing the schema as part of the data.
– Another intrinsic feature of an RDF database is the occurrence of a partial schema. In such case, we must define whether the schema will be used or not. In the first case, it could be necessary to "discover" the schema, and just then transform the data. Hence, such an approach could imply the use of a transformation method that is schema independent, or a combined method that supports data with or without schema.
– A semantic issue is the right and complete interpretation of a reified triple, and its representation in a property graph.
– RDF Schema supports the definition of subclass and subproperty. These features are not supported by current property graph database systems.
– OWL, that is intended to be a layer above RDF Schema, supports more complex constraints for classes (e.g. intersection) and properties (e.g. transitivity). These features are not supported by the property graph model.
– An RDF database could contain semantic information that allows data inference (i.e. to infer new triples based on the existing triples). Current graph database systems have been not designed to support inference.
– Discovering semantic information and resolving mismatches requires the application of human intelligence and judgment. Hence, the semantic interoperability is determined by the power of the translation methods to support data and semantics interpretation.

4.3 Query interoperability

– Unlike the standardisation (via the W3C standards and ISO committees) of query languages for the relational databases (SQL) and the RDF databases (SPARQL), property graph databases do not have a standard query language. This has led to the development of a wide range of vendor-specific graph query languages (e.g. Cypher for Neo4j and Gremlin for Apache TinkerPop).

- Most of the current property graph query languages do not have a solid formal foundation (semantics, complexity and expressiveness). This raises a critical challenge for supporting query interoperability, since a formal mapping between SPARQL and a property graph language cannot be defined.
- The notion of schema in the context of property graph query languages is not strictly defined, or even absent in some cases due to their NoSQL oriented nature. This creates another challenge when aiming to transform RDF data (which consists of schema information) to property graph data.
- Property graph query languages address two different paradigms: *declarative* and *imperative*. For instance, Cypher is a declarative query language, whereas Gremlin is an imperative graph traversal language that also offers a declarative construct. This adds an additional challenge, since these two different paradigms operate on disparate sets of semantics (i.e. set vs bag semantics), while aiming to support query interoperability.
- There are some on-going efforts, such as [41,43], that advocate consolidating the relational and graph algebras in order to lay a foundation for proving the equivalences between the different transformations and mappings to support *query interoperability* between RDF and Property graphs. Nonetheless, there is still scope for improvement.

Therefore, there is a need to propose a standardized query language for property graph databases. It will facilitate the formal definition and study of query transformation methods.

5 Conclusions

Interoperability is a very important feature that should be supported by any database systems. In this article we concentrate on the interoperability between RDF databases and property graph databases. Our analysis of the available approaches and methods does not cover all the issues and challenges, but shows that there are several problems to deal with.

The interoperability among systems is based on agreements between requesters and providers [22]. Hence, the research on the area must be supported by the development of successful standards (starting with the standardization of the property graph data model and its query language).

Acknowledgments. Renzo Angles was (partially) funded by the Millennium Institute for Foundational Research on Data (IMFD). Harsh Thakkar was funded by the EU H2020 R&I project BOOST4.0 (GA 780732). Dominik Tomaszuk was supported by the National Science Center, Poland (NCN) under research grant Miniatura 2.

References

1. Cypher query language reference, version 9. https://s3.amazonaws.com/artifacts.opencypher.org/openCypher9.pdf
2. The GQL Manifesto. https://gql.today/
3. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. IEEE Std 610 (Jan 1991)
4. Angles, R., Arenas, M., Barceló, P., et al.: Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR) 50(5) (Sept 2017)
5. Angles, R., Boncz, P., Larriba-Pey, J., Fundulaki, I., Neumann, T., Erling, O., Neubauer, P., Martinez-Bazan, N., Kotsev, V., Toma, I.: The Linked Data Benchmark Council: a Graph and RDF industry benchmarking effort. Sigmod Record 43(1) (March 2014)
6. Angles, R., Gutierrez, C.: An introduction to graph data management. In: Graph Data Management, chap. 1. Data-Centric Systems and Applications, Springer Nature (2018)
7. Angles, R.: The property graph database model. In: Proc. Alberto Mendelzon International Workshop on Foundations of Data Management (AMW) (2018)
8. Angles, R., Arenas, M., Barceló, P., Boncz, P., Fletcher, G., Gutierrez, C., Lindaaker, T., Paradies, M., Plantikow, S., Sequeda, J., van Rest, O., Voigt, H.: G-CORE: A core for future graph query languages. In: Proc. of the International Conference on Management of Data (SIGMOD) (2018)
9. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: Lightweight Linked Data Publication from Relational Databases. In: Proc. of the 18th International Conference on World Wide Web. pp. 621–630. ACM, New York, NY, USA (2009)
10. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. Journal of Data Semantics 1(3) (2012)
11. Boneva, I., Gayo, J.E.L., Hym, S., Prud'hommeau, E.G., Solbrig, H.R., Staworko, S.: Validating RDF with shape expressions. CoRR, abs/1404.1270 (2014)
12. Bonifati, A., Fletcher, G., Voigt, H., Yakovets, N.: Querying Graphs. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2018)
13. Brickley, D., Guha, R.V.: RDF Schema 1.1, W3C Recommendation (2014)
14. Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL) - W3C Recommendation. https://www.w3.org/TR/grddl/ (September 11 2007)
15. Corby, O., Faron-Zucker, C.: Sttl: A sparql-based transformation language for rdf. In: Proc. of the 11th International Conference on Web Information Systems and Technologies (WEBIST) (2015)
16. Das, S., Srinivasan, J., et al.: A tale of two graphs: Property graphs as RDF in oracle. In: Proc. of the International Conference on Extending Database Technology (EDBT). pp. 762–773 (2014)
17. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: RML: A generic language for integrated rdf mappings of heterogeneous data. In: Proc. of the Linked Data on the Web Workshop (2014)
18. Group, W.O.W.: OWL 2 Web Ontology Language, Document Overview - W3C Recommendation. https://www.w3.org/TR/owl2-overview/ (December 11 2012)
19. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language - W3C Recommendation. https://www.w3.org/TR/sparql11-query/ (March 21 2013)
20. Hartig, O.: Reconciliation of RDF* and property graphs. arXiv:1409.3288 (2014)

21. Hartig, O., Thompson, B.: Foundations of an alternative approach to reification in rdf. arXiv preprint arXiv:1406.3399 (2014)
22. Heiler, S.: Semantic interoperability. ACM Comput. Surv. 27(2), 271–273 (1995)
23. Joundrey, D.N., Taylor, A.G.: The Organization of Information. Library and Information Science, Libraries Unlimited, fourth edn. (2017)
24. Klyne, G., Carroll, J.: Resource Description Framework (RDF) Concepts and Abstract Syntax. http://www.w3.org/TR/2004/REC-115-concepts-20040210/ (February 2004)
25. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL) - W3C Recommendation. https://www.w3.org/TR/shacl/ (July 20 2017)
26. Matsumoto, S., Yamanaka, R., Chiba, H.: Mapping RDF graphs to property graphs. In: Proc. of the Fifth International Workshop on Practical Application of Ontology for Semantic Data Engineering (2018)
27. Nguyen, V., Leeka, J., et al.: A formal graph model for rdf and its implementation. arXiv:1606.00480 (2016)
28. Ouksel, A.M., Sheth, A.: Semantic interoperability in global information systems. SIGMOD Rec. 28(1), 5–12 (Mar 1999)
29. Parent, C., Spaccapietra, S.: Issues and approaches of database integration. Commun. ACM 41(5), 166–178 (May 1998)
30. Parent, C., Spaccapietra, S.: Database integration: the key to data interoperability. Advances in Object-Oriented Data Modeling (2000)
31. Park, J., Ram, S.: Information systems interoperability: What lies beneath? ACM Transactions on Information Systems 22(4), 595–632 (Oct 2004)
32. Pokorný, J., Valenta, M., et al.: Integrity constraints in graph databases. Procedia Computer Science 109, 975–981 (2017)
33. Predoiu, L., Zhdanova, A.V.: Semantic Web Languages and Ontologies, p. 7. Encyclopedia of Internet Technologies and Applications, IGI Global (2008)
34. Prud'hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: An RDF validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems (SEM). pp. 32–40. ACM, New York, NY, USA (2014)
35. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF - W3C Recommendation. https://www.w3.org/TR/rdf-sparql-query/ (January 15 2008)
36. van Rest, O., Hong, S., Kim, J., Meng, X., Chafi, H.: PGQL: a Property Graph Query Language. In: Proc. of the Int. Workshop on Graph Data Management Experiences and Systems (GRADES) (2013)
37. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media, first edn. (June 2013)
38. S.Ceri, Tanca, L., Zicari, R.: Supporting interoperability between new database languages. In: Proc. of the 5th Annual European Computer Conference (CompEuro) (1991)
39. Schätzle, A., Przyjaciel-Zablocki, M., Berberich, T., Lausen, G.: S2X: Graph-parallel querying of RDF with GraphX. In: Biomedical Data Management and Graph Online Querying. pp. 155–168. Springer International Publishing (2016)
40. Sheth, A.P.: Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics, pp. 5–29. Springer US (1999)
41. Szárnyas, G., Marton, J., Maginecz, J., Varró, D.: Reducing property graph queries to relational algebra for incremental view maintenance. arXiv preprint arXiv:1806.07344 (2018)

42. Tandy, J., Herman, I., Kellogg, G.: Generating RDF from Tabular Data on the Web - W3C Recommendation. https://www.w3.org/TR/csv2rdf/ (December 17 2015)
43. Thakkar, H., Punjani, D., Auer, S., Vidal, M.E.: Towards an integrated graph algebra for graph pattern matching with Gremlin. In: International Conference on Database and Expert Systems Applications. pp. 81–91. Springer (2017)
44. Thakkar, H., Punjani, D., Lehmann, J., Auer, S.: Two for one: Querying property graph databases using SPARQL via gremlinator. In: Proc. of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). pp. 1–5. ACM (2018)
45. Thakkar, H., Punjani, D., et al.: A stitch in time saves nine–SPARQL querying of property graphs using gremlin traversals. arXiv:1801.02911 (2018)
46. Tomaszuk, D.: RDF Data in Property Graph Model. In: Proc. of the 10th International Conference on Metadata and Semantics Research (MTSR). vol. 672 (2016)
47. Tomaszuk, D., Angles, R., Szeremeta, L., Litman, K., Cisterna, D.: Serialization for property graphs. In: Proc. of the 15th International Conference Beyond Databases, Architectures and Structures (BDAS) (2019)
48. Zhan, J., Luk, W.S., Wong, C.: An Object-Oriented Approach to Query Interoperability, pp. 141–153. Springer US (1996)