G2GML: Graph to Graph Mapping Language for Bridging RDF and Property Graphs

Hirokazu Chiba[1], Ryota Yamanaka[2], and Shota Matsumoto[3]

[1] Database Center for Life Science, Chiba 277-0871, Japan
chiba@dbcls.rois.ac.jp
[2] Oracle Corporation, Bangkok 10500, Thailand
ryota.yamanaka@oracle.com
[3] Lifematics Inc., Tokyo 101-0041, Japan
shota.matsumoto@lifematics.co.jp

Abstract. How can we maximize the value of accumulated RDF data? Whereas the RDF data can be queried using the SPARQL language, even the SPARQL-based operation has a limitation in implementing traversal or analytical algorithms. Recently, a variety of database implementations dedicated to analyses on the property graph (PG) model have emerged. Importing RDF datasets into these graph analysis engines provides access to the accumulated datasets through various application interfaces. However, the RDF model and the PG model are not interoperable. Here, we developed a framework based on the Graph to Graph Mapping Language (G2GML) for mapping RDF graphs to PGs to make the most of accumulated RDF data. Using this framework, accumulated graph data described in the RDF model can be converted to the PG model, which can then be loaded to graph database engines for further analysis. This study bridges RDF and PGs and contributes to interoperable management of knowledge graphs, thereby expanding the use cases of accumulated RDF data. Demonstration of the G2G mapping framework is available at `https://purl.org/g2gml`.

Keywords: RDF · Property Graph · Graph Database

1 Introduction

Increasing amounts of scientific and social data are being published in the form of the Resource Description Framework (RDF), which presently constitutes a large open data cloud. DBpedia [1] and Wikidata [2] are well-known examples of such RDF datasets. SPARQL is a protocol and query language that serves as a standardized interface for RDF data. This standardized data model and interface enables the construction of integrated graph data. However, the lack of an interface for graph-based analysis and performant traversal limits use cases of the graph data.

Recently, the property graph (PG) model [3,4] has been increasingly attracting attention in the context of graph analysis. Various graph database engines, including Neo4j [5], Oracle Database [6], and Amazon Neptune [7] adopt this model. These graph database engines support algorithms for traversing or analyzing graphs. However, few datasets are published in the PG model and the lack of an ecosystem for exchanging data in the PG model limits the application of these powerful engines.

In the light of this situation, developing a method to transform RDF into PG would be highly valuable. One of the practical issues faced by this challenge is the lack of a standardized PG model. Another issue is that the transformation between RDF and PG is not straightforward due to the differences in their models. In RDF graphs, all information is expressed by triples (node-edge-node), whereas in PGs, arbitrary information can be contained in each node and edge as the key-value form. Although this issue was previously addressed on the basis of predefined transformations [8], users still cannot control the mapping for their specific use cases.

In this study[9][4], we redefine the PG model incorporating the differences in existing models and propose serialization formats based on the data model. We further propose a graph to graph mapping framework based on the Graph to Graph Mapping Language (G2GML). Employing this mapping framework, accumulated graph data described in RDF can be converted into PGs, which can then be loaded into several graph database engines for further analysis.

2 Overview

We provide an overview of the graph to graph mapping (G2G mapping) framework (Figure 1).

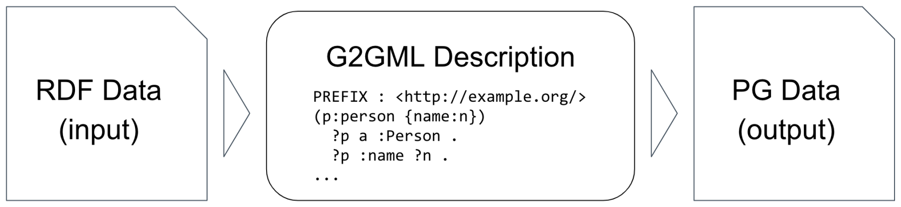
In this framework, users describe mappings from RDF to PG in G2GML. According to this G2GML description, the input RDF dataset is converted into a PG dataset. The new dataset can also be optionally saved in specific formats for loading into major graph database implementations.

G2GML is a declarative language comprising pairs of RDF patterns and PG patterns. The core concept of a G2GML description can be represented by a map from RDF subgraphs, which match specified SPARQL patterns, to PG components.

3 Examples

Figure 3 shows the minimal example of G2G mapping from RDF data (Figure 2) to PG data (Figure 4), representing the following five types of typical mapping. Here, this PG data is represented in a general PG format we defined previously [10].

[4] Main paper to be presented at ISWC 2020



```
PREFIX : <http://example.org/>
(p:person {name:n})
  ?p a :Person .
  ?p :name ?n .
...
```

Fig. 1. Overview of mapping from RDF to PG

```
@prefix : <http://example.org/> .
:person1 a :Person ;
        :name 'Alice' .
:person2 a :Person ;
        :name 'Bob' .
:person1 :supervised_by :person2 .
[] a :Email ;
   :sender :person1 ;
   :receiver :person2 ;
   :year 2017 ;
   :attachment '01.pdf' .
```

Fig. 2. Example of input RDF data

```
 1 PREFIX : <http://example.org/>
 2 (p:person {name:n})
 3   ?p a :Person .
 4   ?p :name ?n .
 5 (p1:person)-[:supervised_by]->(p2:person)
 6   ?p1 :supervised_by ?p2 .
 7 (p1:person)-[:emailed {year:y, attachment:a}]->(p2:person)
 8   ?f a :Email ;
 9      :sender ?p1 ;
10      :receiver ?p2 ;
11      :year ?y .
12   OPTIONAL { ?f :attachment ?a }
```

Fig. 3. Example of G2G mapping definition

– Resource to node: In lines 2–4, the RDF resources with type `:Person` are mapped into the PG nodes using their IRIs as node IDs.
– Datatype property to node property: In lines 2–4, the RDF datatype property `:name` is mapped onto the PG node property key `name`. The literal objects `'Alice'` and `'Bob'` are mapped onto the node property values.
– Object property to edge: In lines 5–6, the RDF object property `:supervised_by` is mapped onto the PG edge `supervised_by`.
– Resource to edge: In lines 7–12, the RDF resource with type `:Email` is mapped onto the PG edge `emailed`.
– Datatype property to edge property: In lines 7–12, the RDF datatype property `:year` and `:attachment` are mapped onto the PG edge property `year` and `attachment`. The literal objects `2017` and `'01.pdf'` are mapped onto the edge property values.

```
"http://example.org/person1" :person name:Alice
"http://example.org/person2" :person name:Bob
"http://example.org/person1" -> "http://example.org/person2" :supervised_by
"http://example.org/person1" -> "http://example.org/person2" :emailed year:2017 attachment:"01.pdf"
```

Fig. 4. Example of output PG data

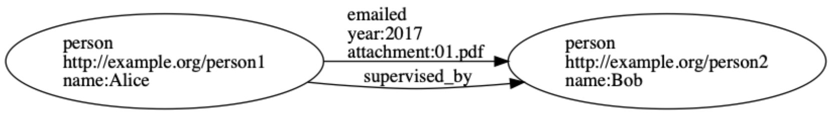


Fig. 5. Example of output PG data (Visualization)

4 Related Work

A preceding study on converting existing data into graph data included an effort to convert relational databases into graph databases [11]. However, given that RDF has prevailed as a standardized data model in scientific communities, considering mapping based on the RDF model is crucial. The interoperability of RDF and PG [8,12,13,14] has been discussed, and efforts were made to develop methods to convert RDF into PG [15,16]. However, considering the flexibility regarding the type of information that can be expressed by edges in property graphs, a novel method for controlling the mapping is necessary.

To the best of our knowledge, this study presents the first attempt to develop a framework for controlled mapping between RDF and PG. Notably, the designed G2GML is a declarative mapping language. As a merit of the declarative description, we can concentrate on the core logic of mappings. In the sense that the mapping process generates new graph data on the basis of existing graph data, it has a close relation to the semantic inference.

Other mapping frameworks, such as Neosemantics (a Neo4j plugins), propose a method to convert RDF datasets without mapping definitions. We observe a similar discussion in the conversion from the relational model to RDF, where are two W3C standards, i.e., Direct Mapping [17] and R2RML [18].

5 Availability

The prototype implementation of G2G mapping is available on GitHub (`https://github.com/g2glab/g2g`) under MIT license, which is written in JavaScript and can be executed using Node.js in the command line. It has an endpoint mode and a local file mode. The local file mode uses Apache Jena ARQ to execute SPARQL queries internally, while the endpoint mode accesses SPARQL endpoints via the Internet. An example of the usage in the endpoint mode is as follows:

```
$ g2g musician.g2g http://dbpedia.org/sparql
```

where the first argument is a G2GML description file, and the second argument is the target SPARQL endpoint, which provides the source RDF dataset.

Furthermore, a demonstration site (`https://purl.org/g2gml`) is available, and the documentation (`https://g2gml.readthedocs.io`) also includes quick tutorials on how to try G2GML using the Docker image (`https://hub.docker.com/r/g2glab/g2g`).

References

1. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., ... Bizer, C.: DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 6(2), 167-195. (2015)
2. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM, 57(10), 78-85. (2014)
3. Angles, R., Gutierrez, C.: An introduction to Graph Data Management. arXiv preprint arXiv:1801.00036 (2017)
4. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoc, D.: Foundations of Modern Query Languages for Graph Databases. ACM Computing Surveys (CSUR), 50(5), 68. (2017)
5. The Neo4j Graph Platform. `https://neo4j.com/`
6. Oracle Database Property Graph. `https://www.oracle.com/goto/propertygraph`
7. Amazon Neptune. `https://aws.amazon.com/neptune/`
8. Hartig, O.: Reconciliation of RDF* and property graphs. arXiv preprint arXiv:1409.3288 (2014)
9. Chiba, H., Yamanaka, R., Matsumoto, S.: G2GML: Graph to Graph Mapping Language for Bridging RDF and Property Graphs. Proceedings of the 19th International Semantic Web Conference (2020)
10. Chiba, H., Yamanaka, R., Matsumoto, S. Property Graph Exchange Format. arXiv preprint arXiv:1907.03936 (2019)
11. De Virgilio, R., Maccioni, A., Torlone, R.: Converting relational to graph databases. In First International Workshop on Graph Data Management Experiences and Systems, p. 1. ACM (2013)
12. Angles, R., Thakkar, H., Tomaszuk, D.: RDF and Property Graphs Interoperability: Status and Issues. Proceedings of the 13th Alberto Mendelzon International Workshop on Foundations of Data Management (2019)
13. Das, S., Srinivasan, J., Perry, M., Chong, E. I., Banerjee, J.: A Tale of Two Graphs: Property Graphs as RDF in Oracle. In EDBT, pp. 762–773 (2014)
14. Thakkar, H., Punjani, D., Keswani, Y., Lehmann, J., Auer, S.: A Stitch in Time Saves Nine–SPARQL querying of Property Graphs using Gremlin Traversals. arXiv preprint arXiv:1801.02911 (2018)
15. Tomaszuk, D.: RDF data in property graph model. In Research Conference on Metadata and Semantics Research, pp. 104–115 (2016)
16. De Virgilio, R.: Smart RDF data storage in graph databases. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 872–881. IEEE (2017)
17. A Direct Mapping of Relational Data to RDF, W3C Recommendation 27 September 2012 `https://www.w3.org/TR/r2rml/`
18. R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012 `https://www.w3.org/TR/r2rml/`