Mapping RDF Graphs to Property Graphs

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

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

Abstract. Increasing amounts of scientific and social data are published in the Resource Description Framework (RDF). Although 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 graph database implementations dedicated to analyses on the property graph model have emerged. However, the RDF model and the property graph model are not interoperable. Here, we developed a framework based on the Graph to Graph Mapping Language (G2GML) for mapping RDF graphs to property graphs to make the most of accumulated RDF data. Using this framework, graph data described in the RDF model can be converted to the property graph model and can be loaded to several graph database engines for further analysis. Future works include implementing and utilizing graph algorithms to make the most of the accumulated data in various analytical engines.

Keywords: RDF · Property Graph · Graph Database

1 Introduction

Increasing amounts of scientific and social data are described as graphs. As a format of graph data, the Resource Description Framework (RDF) is widely used. Although RDF data can be queried using the SPARQL language in a flexible way, SPARQL is not dedicated to traversal of graphs and has a limitation in implementing graph analysis algorithms.

In the context of graph analysis, the property graph model [1] is becoming popular; various graph database engines, including Neo4j [2], Oracle Labs PGX [3], and Amazon Neptune [4], adopt this model. These graph database engines support algorithms for traversal or analyzing graphs. However, currently not many datasets are consistently described in the property graph model, so the application of these powerful engines are limited.

Considering this situation, it is valuable to develop a method to transform RDF data into property graphs. However, the transformation is not straightforward due to the differences in the data model. In RDF graphs, all information is

expressed as the triple (node-edge-node), whereas in property graphs, arbitrary information can be contained in each of the nodes and edges as key-value form. Although previous works addressed this issue by formalizing transformations [5], users cannot define their specific mappings intended for each use case.

Here, we developed a framework based on the Graph to Graph Mapping Language (G2GML) for mapping RDF graphs to property graphs. Using this framework, accumulated graph data described in the RDF model can be converted to the property graph model and can be loaded to several graph database engines.

2   Methods

Figure 1 shows the overview of proposed framework. In the proposed framework, users write mappings from RDF graphs to property graphs in G2GML. This mapping can be processed by an implementation called *G2G Mapper*, which is implemented by authors (available on `https://github.com/g2gml`). This tool retrieves RDF data from SPARQL endpoints and converts them to property graph data in several different formats specified by popular graph databases.

G2GML is a declarative language which consists of pairs of RDF graph patterns and property graph patterns. An intuitive meaning of a G2GML is a mapping between RDF subgraphs that matches the described patterns and described components of the property graph. In the next section, we briefly explain the syntax of G2GML with a concrete example usage.
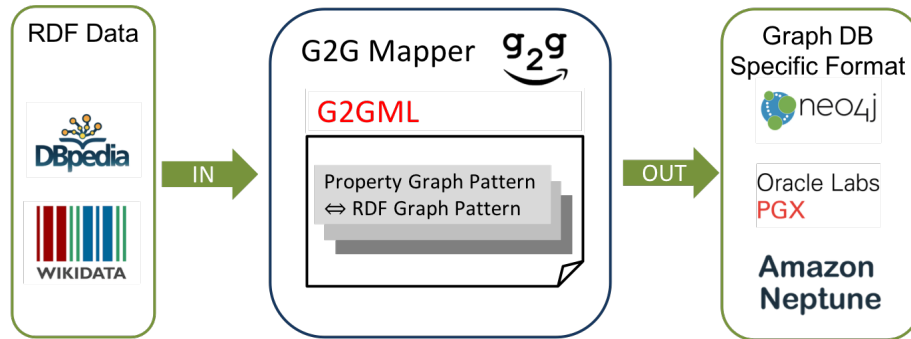

Fig. 1. Overview of G2GML mapping

3   Example

Figure 2 shows an example of G2GML mapping, which converts RDF data retrieved from DBpedia into property graph data. When we focus on relationships
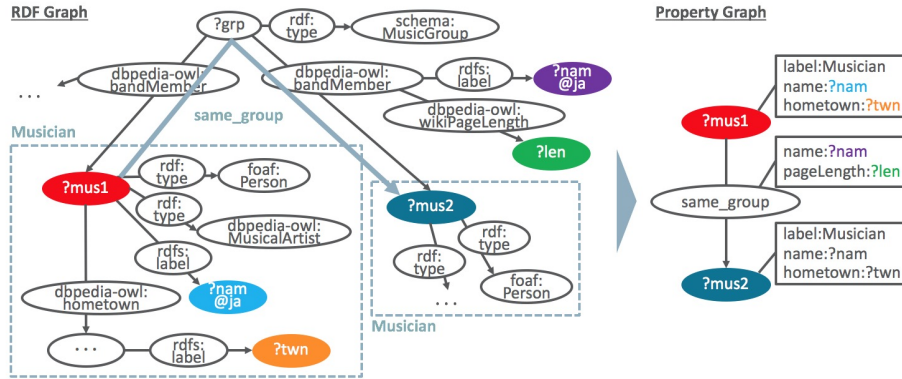
Fig. 2. Mapping of RDF data

that one musician and another are in the same group, the information can be summarized into the property graph data as shown in this figure.

For this conversion, the actual G2GML is described as in Figure 3. It starts with URI prefixes used to write mappings, and then, each mapping consists of one unindented line of a property graph pattern and indented lines of an RDF graph pattern. A property graph pattern is written in a syntax like Cypher (the query language of Neo4j), whereas an RDF graph pattern is written as a pattern in SPARQL. Variables in each pattern are mapped by those names. This example contains one node mapping for `Musician` entity and one edge mapping for `same_group` relationship only. In G2GML, edge mappings are defined based on the conditions of node mappings, which means that edges are generated in property graph iff both nodes' patterns and edges' patterns are matched in RDF graph. Also, `mus, nam, dat, twn` and `len` are used as variables to extract resources and literals from RDF graph. In the resulting property graph, resources can be mapped to nodes, while literals can be mapped to values of properties.

Finally, Figure 4 shows the SPARQL query to retrieve the pairs of musicians who are in the same group. After G2GML mapping above, we can load the generated property graph data into graph databases, such as Oracle Labs PGX, and the query can be written in PGQL (the query language of PGX).

4   Conclusion

In this work, we defined G2GML for mapping RDF graphs to property graphs and implemented a converter based on the G2GML. We also showed an example usage of G2GML. Future works include further analysis of the converted graph data on the database engines adopting the property graph model.

```
# Prefixes
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX prop: <http://dbpedia.org/property/>
PREFIX schema: <http://schema.org/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

# Node mapping
(mus:Musician {vis_label:nam, born:dat, hometown:twn})              # PG Pattern
    ?mus rdf:type foaf:Person, dbpedia-owl:MusicalArtist .          # RDF Pattern
    ?mus rdfs:label ?nam .
    OPTIONAL { ?mus prop:born ?dat }
    OPTIONAL { ?mus dbpedia-owl:hometown / rdfs:label ?twn }

# Edge mapping
(mus1:Musician)-[:same_group {label:nam, length:len}]->(mus2:Musician)   # PG Pattern
    ?grp a schema:MusicGroup ;                                      # RDF Pattern
        dbpedia-owl:bandMember ?mus1 , ?mus2 .
    FILTER(?mus1 != ?mus2)
    OPTIONAL { ?grp rdfs:label ?nam. FILTER(lang(?nam) = "ja")}
    OPTIONAL { ?grp dbpedia-owl:wikiPageLength ?len }
```

Fig. 3. G2GML mapping definition

```
# SPARQL
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT
    ?nam1 ?nam2
WHERE {
    ?mus1 rdf:type foaf:Person , dbpedia-owl:MusicalArtist .
    ?mus2 rdf:type foaf:Person , dbpedia-owl:MusicalArtist .
    ?mus1 rdfs:label ?nam1 . FILTER(lang(?nam1) = "ja") .
    ?mus1 rdfs:label ?nam2 . FILTER(lang(?nam2) = "ja") .
    ?grp a schema:MusicGroup ;
        dbpedia-owl:bandMember ?mus1 , ?mus2 .
    FILTER(?mus1 != ?mus2)
}

# PGQL
SELECT DISTINCT m1.name, m2.name WHERE (m1)-[same_group]-(m2)
```

Fig. 4. SPARQL and PGQL

References

1. Angles, R., Arenas, M., Barcel, P., Hogan, A., Reutter, J., Vrgoc, D.: Foundations of Modern Query Languages for Graph Databases. ACM Computing Surveys (CSUR), 50(5), 68 (2017)
2. The Neo4j Graph Platform, https://neo4j.com/.
3. Oracle Labs Parallel Graph AnalytiX (PGX), https://www.oracle.com/technetwork/oracle-labs/parallel-graph-analytix/overview/index.html
4. Amazon Neptune, https://aws.amazon.com/neptune/.
5. Hartig, O.: Reconciliation of RDF* and property graphs. arXiv preprint arXiv:1409.3288 (2014)