Graph-Based Editing of Linked Data Mappings Using the RMLEditor

Pieter Heyvaert(✉), Anastasia Dimou, Ruben Verborgh, Erik Mannens, and Rik Van de Walle

Data Science Lab, Ghent University - IMinds, Ghent, Belgium
pheyvaer.heyvaert@ugent.be

Abstract. Linked Data is in many cases generated from (semi-) structured data. This generation is supported by several tools, a number of which use a mapping language to facilitate the Linked Data generation. However, knowledge of this language and other used technologies is required to use the tools, limiting their adoption by non-Semantic Web experts. We demonstrate the RMLEditor: a graphical user interface that utilizes graphs to easily visualize the mappings that deliver the RDF representation of the original data. The required amount of knowledge of the underlying mapping language and the used technologies is kept to a minimum. The RMLEditor lowers the barriers to create Linked Data by aiming to also facilitate the editing of mappings by non-experts.

1 Introduction

Linked Data [1] is one of the most important aspects that drives the adoption of Semantic Web technologies, as it interlinks data whose semantically enriched representation is available. Most of the current Linked Data stems originally from data in (semi-)structured formats. Mappings languages, such as R2RML [2] and RML [3], specify in a declarative way how Linked Data is generated from such (semi-)structured data. This data is possessed by data specialists, who are not Semantic Web experts or developers. Therefore, data specialists should be able to specify the mappings, modify and extend them at any time, while limiting the awareness of the underlying mapping languages and technologies.

Nevertheless, dedicated environments that support users to intuitively edit mappings were not thoroughly investigated yet, and each have their limitations. First, *step-by-step* wizards, e.g., the fluidOps editor[1], prevailed as an easy-to-reach solution. However, such applications restrict data publishers' editing options, hamper altering parameters in previous steps, and detach mapping definitions from the overall knowledge modeling, since related information is separated in different steps. Second, a number of tools, such as the fluidsOps editor,

The described research activities were funded by Ghent University, iMinds, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO Flanders), and the European Union.

[1] http://www.fluidops.com.

H. Sack et al. (Eds.): ESWC 2016 Satellite Events, LNCS 9989, pp. 123–127, 2016.
DOI: 10.1007/978-3-319-47602-5_25

sheet2RDF[2] and -ontoPro-[3], limit the user to a specific data format that can be used to generate Linked Data. This limitation makes integration of data distributed across data sources in different data formats impossible. Though, other tools, e.g., Karma[4], DataOps[5] and RDF123[6] do support heterogeneous data sources. Nevertheless, third, all aforementioned tools require users to understand the mapping language's syntax, as it is widely used in their graphical user interface (GUI). Therefore, data specialist are only able to create their own mappings with the help of Semantic Web experts or by acquiring the required knowledge of the language themselves.

We propose a demo of the RMLEditor[7], an editing environment for specifying mappings of (semi-)structured data to their RDF representation based on graph visualizations, that does not suffer from the aforementioned limitations. The demo accompanies a paper in the in-use track of the ESWC2016 conference [4]. Participants are able to perform their own mappings, on multiple data sources in different data formats, using a live instance of the RMLEditor, as shown via the following screencast that is available at https://www.youtube.com/watch?v=J7OtSYnZD9I.

2 RMLEditor

A mapping editor is an application that allows users to describe how Linked Data is generated based on (semi-)structured data, including tabular data (e.g., CSV) and hierarchical data (e.g., XML and JSON). It should have a GUI that is understandable and usable by non-Semantic Web experts. To achieve this, a list of seven desired features were defined in previous work [5]. The RMLEditor covers all of them:

1. The RMLEditor is independent of the underlying mapping language, so users are able to create mappings with a limited amount of knowledge of the language's syntax.
2. It allows users to execute the mappings outside of the editor, because the editor is mainly meant to edit the mappings. Users can export the mappings and execute them using different tools.
3. It enables users to map multiple data sources at the same time, as it might occur that data that is required to describe a knowledge domain is spread across multiple sources.
4. It supports data sources in different data formats, e.g., CSV, and XML, as the Linked Data is independent of the data's original format.
5. As multiple schemas (ontologies and vocabularies) can be used to create a mapping, it supports the use of both existing and customs schemas.

[2] http://art.uniroma2.it/sheet2rdf/.
[3] http://ontop.inf.unibz.it/components/sample-page/.
[4] http://usc-isi-i2.github.io/karma/.
[5] http://www.fluidops.com.
[6] http://ebiquity.umbc.edu/project/html/id/82/RDF123.
[7] http://rml.io/RMLeditor; licensing options available on request.

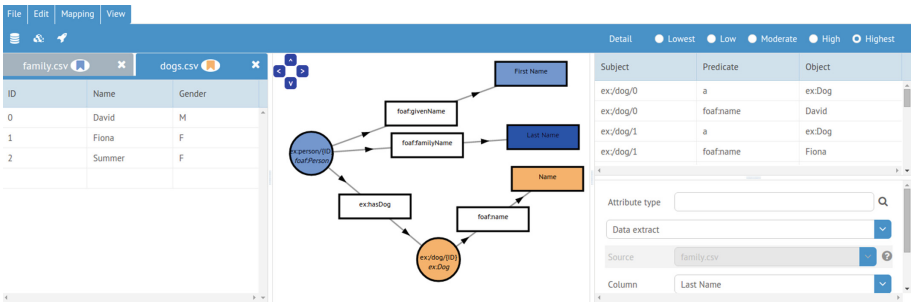

Fig. 1. The RMLEditor with the *Input Panel* on the left, the *Modeling Panel* in the center and the *Results Panel* on the right

6. It allows multiple alternative modeling approaches [6], as certain use cases might benefit from using a specific approach.
7. By supporting non-linear workflows, users are able to keep an overview of the mapping model and its relationships.

The RMLEditor is an application available in the browser. The mappings are visualized in the GUI by using graphs. Users can create a new or upload an existing mapping. Creating, updating and extending the mappings is done by performing the corresponding graph manipulations. The RMLEditor triggers the mapping processor which executes the mappings, exported by the RMLEditor, and generates RDF statements. We chose the RML as the RMLEditor's underlying mapping language. However, any other mapping language could be used instead, if it allows to implement the features, which is the case for RML.

3 Graphical User Interface

One of the most important aspects of a mapping editor is its GUI, as indicated by the aforementioned features. In the RMLEditor, we offer three panels, that implement these features, to the users: *Input Panel*, *Modeling Panel* and *Results Panel* (see Fig. 1).

In the following, we elaborate on how the features are implemented in the RMLEditor. The first feature is implemented via the *ModelingPanel*, as this panel offers a generic representation of the mappings, independent of the underlying mapping language, by using a graph representation. Mappings are created and edited by manipulating the nodes and edges. The second feature is also implemented via this panel, as it allows to export both the graph representation and the RML statements. This allows to execute the mappings outside the RMLEditor. The third feature is implemented via the *Input Panel*, because in this panel users are able to view the different data sources. Each data source can be uniquely identified by its color that is automatically and arbitrarily assigned by the RMLEditor. Additionally, the nodes' and edges' colors depend on the data source that is used in that specific mapping. The fourth feature is implemented

by choosing an adequate visualization for each data source based on the data format. The fifth feature is facilitated by allowing users, through manipulations on the graph, to add semantic annotations using multiple schemas.

In previous work [6] we described the different mapping generation approaches. The *data-driven* approach uses the input data sources as the basis to construct the mappings. The classes, properties and datatypes of the schemas are then assigned to the mappings. When users start with the schemas to generate the mappings, the *schema-driven* approach is followed. Next, data fractions from the data sources can be associated to the mappings. The sixth feature is facilitated via the *Input Panel*, *Modeling Panel* and the *Result Panel*, as their functionality and interaction supports these approaches, as explained in more detail in Sect. 4. The latter panel shows the resulting RDF dataset when the mappings defined in the *Modeling Panel* are executed on the data in the *Input Panel*. For each RDF triple of the dataset it shows the subject, predicate and object. The last feature is facilitated by allowing the users to decide which panel they use and at what moment in the mapping process they use it. In linear workflows, this is not the case. Additionally, the panels are aligned next to each other, and when users want to focus on a specific panel, they are able to hide the other panels.

4 Editing Mappings

For Linked Data, an entity is one of the most important aspects. An *entity is something in the world, identified by a unique name (URI)*. Anything can be an entity, including physical things, documents and abstract concepts. A URI for a person named 'John Doe' could be http://www.example.com/john_doe. We assume for this example that every person's name is unique. Therefore, using the name in the URI results in a unique URI for each person. Additionally, the use of this functional attribute ensures that always the same URI is generated between different mapping executions. The classes defined in schemas are used to define the type of an entity, e.g. `foaf:Person` (where `foaf:` is expanded to the full URI of the vocabulary http://xmlns.com/foaf/0.1/). When no URI is provided for an entity, we call it a blank node. Information about an entity is represented by both *attributes* and *relationships*. Examples of attributes are 'John Doe' (name of a person) and 'BE0596.342.234' (VAT number of a company). Relationships connect entities and attributes. For example, the relationship with property `foaf:name` states that the string 'John Doe' (attribute) is the name of http://www.example.com/john_doe (entity).

Specifying entities, attributes and relationships is what composes the creation of mappings. This can be done using the two aforementioned approaches. When following the data-driven approach, users first load all the data in the RMLEditor. Next, they create entities and attributes based on the different data fractions, by interacting with the corresponding elements in the *Input Panel*. If blank nodes are required, they are created via the *Modeling Panel*. Users define the classes of the entities and blank nodes, and the datatypes of the attributes. This is done

by clicking on a node, which brings up a panel where the node's details can be edited. Additionally, for the entities they define how the URIs are generated. Users also need to define relationships to connect entities with their attributes or other entities, together with selecting the correct property from the schemas. This is done in the same way as with nodes. The Linked Open Vocabularies[8] can be consulted via the GUI to get suggestions on which classes, properties and datatypes to use. Finally, the mapping is executed, and the resulting RDF triples are visible in the *Results Panel*. This last step can also be performed earlier on when not all data fractions are mapped. This allows users to inspect the RDF triples before the mapping is complete, which makes it possible to fix errors earlier on in the process.

When following the schema-driven approach, users mostly interact with the *Modeling Panel*. Through this panel, they can create entities, blank nodes and attributes, based on one or multiple schemas. Relationships are added, based on the properties defined in the schemas. Up until now no references to data sources are made. Therefore, the relevant data sources are loaded and they appear in the *Input Panel*. Every mapping definition is updated to incorporate the use of the data sources where applicable. Finally, again the mapping is executed, and the resulting RDF triples are available in the *Results Panel*.

Regardless of the approach users apply, the RMLEditor assists (non-)Semantic Web experts in editing their Linked Data mappings, while limiting the amount of knowledge needed of RML or the other used technologies. Additionally, facilitating the editing of mappings further lowers the barriers of obtaining Linked Data and thus stimulates the adoption of Semantic Web technologies.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. In: Emerging Concepts, Semantic Services, Interoperability and Web Applications (2009)
2. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. In: Working group recommendation, W3C, September 2012. http://www.w3.org/TR/r2rml/
3. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: Workshop on Linked Data on the Web (2014)
4. Heyvaert, P., Dimou, A., Herregodts, A.-L., Verborgh, R., Schuurman, D., Mannens, E., Van de Walle, R.: RMLEditor: a graph-based mapping editor for linked data mappings. In: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 709–723. Springer, Heidelberg (2016). doi:10.1007/978-3-319-34129-3_43
5. Heyvaert, P., Dimou, A., Verborgh, R., Mannens, E., Van de Walle, R.: Towards a uniform user interface for editing mapping definitions. In: Proceedings of the 4th Workshop on Intelligent Exploration of Semantic Data (2015)
6. Heyvaert, P., Dimou, A., Verborgh, R., Mannens, E., Van de Walle, R.: Approaches for generating mappings to RDF. In: Proceedings of the 14th International Semantic Web Conference: Posters and Demos (2015)

[8] http://lov.okfn.org/dataset/lov/.