RMLx: Mapping Interface for Integrating Open Data with Linked Data Exploration Environment

Peb R. Aryan, Fajar J. Ekaputra,
Elmar Kiesling, A Min Tjoa

Institute of Software Technology and Information Systems,
Technical University of Vienna,
Favoritenstraße 9-11, 1040 Vienna, Austria
{firstname.lastname}@tuwien.ac.at

Kabul Kurniawan

Multimedia Information Systems Group,
University of Vienna,
Währingerstraße 29, 1190 Vienna, Austria
kabulk87@univie.ac.at

Abstract—Recent advances in linked data generation through mapping such as RML (RDF mapping language) allows for providing large-scale RDF data in a more automatic way. However, considerable amount of data in open data portals remain inaccessible as linked data. This is due to the nature of data portals having large number of small-size dataset which makes writing mapping description becomes tedious and error-prone. Moreover, these data sources requires additional preprocessing before To solve this challenge, We introduce extensions to RML to support required tasks and developed RMLx, a visual web-interface to create RML mappings. Using this interface, the process of creating mapping description can become faster and less error-prone. Furthermore, the process of linked data generation can be wrapped as to enable integration with other data in a linked data exploration environment. We explore on four different use cases to identify the requirements followed by describing how these are solved.

I. INTRODUCTION

The recent uptake of open data publishing has led to a large increase of available data sources on the web. Despite the great value inherent to this data, its heterogeneity in terms of format and structure poses a serious challenge that needs to be tackled before its full potential can be leveraged. To solve this problem, mapping-based approaches [1][2] was developed which cover mostly formats used on the web (e.g. XML, JSON, CSV).

The RDF mapping language (RML)[1] enables data publishers to map heterogeneous data to Linked Data, which is a set of best practices for web data publishing. RML mapping description is formulated as RDF which contains descriptive means to map existing data elements to the generated RDF data. However, existing efforts typically do not provide a means to comfortably create mappings and writing mappings manually is often prone to error.

In this work, we propose RMLx, a web-based visual interface to alleviate the definition of mapping descriptions for users. In addition, our experience of mapping data into RDF using RML revealed a necessity to perform transformations which are neither defined in the R2RML, nor in the RML specification. Therefore, we present an extension for doing data transformation and describe the integration of the mapped data for four use cases.

The paper starts by a short review on mapping-based approach of linked data generation in section II followed by identifying the challenges of mapping-based approach. Section III explains the architecture of RMLx and how it address the challenges. Section IV reports on a set of use cases using RMLx approach. We mention related works on Section V and conclude our paper in Section VI.

II. MAPPING-BASED GENERATION OF LINKED DATA

The World Wide Web Consortium (W3C), an international standards organization for the World Wide Web, recommends a standard way to describe mappings of relational data into RDF named R2RML¹. Several approaches are proposed to extend R2RML to support web resources (RML) [1] and non-relational databases (xR-2RML) [2]. In R2RML, data values are identified via column names of relational tables. RML generalize this notion and formalizes a reference, which allows to select data from tabular and hierarchical structures (e.g., JSON, XML).

in the following we identify some challenges that needs to be addressed in order to RML being practical to use:

- **mapping reusability** Publishing data in small portion or slices causes data published in data portals are more likely to have similarity in structure. The reason is to increase specificity of the content and to reduce latency and response time from unneeded data transfer. In this situation, different mapping for each slice might only differ slightly (e.g. in source URL).
- **preprocessing** The content of the published data might not be in the ready-to-use forms. This might be the case since consumers might use the data differently than when it was produced. Data cleaning, extraction of specific parts are some common tasks in preparing the data. One illustrating example to this challenge is when a data value contains multiple elements in a string and the target RDF these values are modeled as separate literal values in different triple.
- **Technical knowledge** RML uses RDF as data model. Although human-friendly serializationx such as Nota-

¹<https://www.w3.org/TR/r2rml/>, Accessed October 19, 2017

Fig. 1: Architecture of RMLx

tion3, Turtle, and JSON-LD are available, writing such description manually for large amount of data is still considered tedious and prone to typing errors. A tool is required to ensure the syntactical errors are minimized and users can focus on writing the correct content.

- **data use** The motivation for publishing linked data is the explicit semantics which makes it easier for machine to understand and to aid in integrating with other data. Thus, getting into RDF is only a means to the end. For users to get benefit, they need to make it usable immediately.

III. RMLx SYSTEM ARCHITECTURE

Challenges mentioned in the previous section is addressed as extension to the existing RML vocabulary. for the proposed extension, defined vocabulary we use `rmlx` prefix². This extension is then implemented in the RML processor which is our own implementation of RML processor. This RML processor is implemented in PHP, which is supported in general for hosting web application.

RMLx system is built around a mapping processor for extended version on RML. The processor lies in the backend and accessible as REST-based web service. From this service we build a form-based interface for constructing the mapping without having to write manually the rdf document. In addition to this service, we also provide a wrapper/adaptor service in order to publish the generated rdf data as data source component (widget) for data mashup environment[3].

Our system, RMLx, is deployed as a web application and is composed of a front-end and back-end module³, as shown in Figure 1. The front-end is written in HTML and JavaScript and the back-end is written in PHP. The footprint of the code is small in comparison to the reference implementation of RML while being able to process small and medium-sized data over HTTP.

Developers can develop RML mapping for the target data sources using RMLx front-end (RMLx Editor) with the assistance of vocabularies from LOV⁴. As an output, users can directly acquire data as RDF or as a data widget -that can be exported from RDFx Editor- in the Linked Widget Platform (LWP⁵).

A. Reference Binding

The namespace defined in RML specification for a reference are limited to the existing column names for tabular data and path names for hierarchical (XML/JSON) data. Our approach extends this namespace by allowing binding to new references apart from what has already available in the data. These bindings are used to store values resulting from data transformations or passing external values as run-time parameters to the mapping process.

Listing 1 shows an example of how new reference are bound to some initial values. New properties for this context are `rmlx:defaultValue` and `rmlx:varName`. The former denotes an relation to an assignment object, and the latter binds new reference name explicitly.

Listing 1: RMLx values binding example

```

<#default> rmlx:defaultValue
[rmlx:varName "refArea";
  rr:constant "http://pebbie.org/ns/wb/
  countries/GB"],
[rmlx:varName "subject"; rr:constant
"http://data.worldbank.org/indicator/SP.POP.
TOTL"],
[rmlx:varName "unit"; rr:constant "person"].

```

One of the usage of this new reference binding mechanism is to allow data source to be parameterized. As illustrated in the Listing 2, data source URL is defined

²short for <http://pebbie.org/ns/rmlx#>

³<http://mashup.pebbie.org/rml>, Accessed October 19, 2017

⁴<http://lov.okfn.org/dataset/lov/>

⁵<http://linkedwidgets.org/>

using `rmlx:sourceTemplate` property. This new property enables the same mapping description to be reused for similarly structured data source. In the following example, `country_code` and `indicator` are not-existent in the content of the data source.

Listing 2: RMLx parameterization example

```
<#metamapping> rmlx:logicalSource[
  rmlx:iterator "$[0]..";
  rmlx:referenceFormulation ql:JSONPath;
  rmlx:sourceTemplate "http://api.worldbank.org
    /countries/{country_code}/indicators/{
      indicator}?format=json&page=1&per_page=1"
  ].
```

B. Data Transformation

We use `rmlx:transform` to denote bindings to new reference created from application of a function. `rmlx:outputVar` is analog to `rmlx:varName` for use as explicit binding to reference name which holds the result of the function. Function is referenced using `rmlx:function` property. The parameters of the function is defined by using `rmlx:parameterMapping` which assigns a value from a value associated with `rml:reference` or `rr:constant` property.

Built-in data transformation functions such as string manipulation (e.g. split, replacement, join) are implemented in the mapping engine which can be used to preprocess the data before being mapped to RDF terms. The Listing 3 shows the example of function application annotation.

Listing 3: RMLx transformation functions example

```
<#default> rmlx:transform [
  rmlx:outputVar "area_part";
  rmlx:function rmlx-op:split;
  rmlx:parameterMapping [rmlx:varName "
    separator";
  rr:constant "/";
  rmlx:parameterMapping [rmlx:varName "input";
  rml:reference "refArea" ]];
rmlx:transform [
  rmlx:outputVar "country";
  rmlx:function rmlx-op:assign;
  rmlx:parameterMapping [rmlx:varName "value";
  rml:reference "area_part[-1]" ]].
```

Aside from explicit reference binding used for the parameter and result of a function, there is also implicit reference binding. This is the case for when a function returns a multi-valued object (e.g. split function returns array of string). In this case, new reference names are generated from existing reference name in `rmlx:outputVar` and array indexing expression `[i]` where `i` are the allowed index for the result. Positive array indices is used to refer array elements starting from first element to the last while negative indices is used to refer last to first elements.

Custom user-defined functions are implemented by associating `rmlx:function` property with REST-based web

service annotation. We use HTTP vocabulary⁶ in conjunction to `rmlx:function` property to describe web service-based function annotation. Such mechanism enables more sophisticated functions such as entity linking/disambiguation by implementing wrapper function from existing services such as DBpedia lookup. An example of definition of user-defined function is shown in the Listing 4.

Listing 4: RMLx user-defined functions example

```
<#CountryDisambiguation> rmlx:transform [
  rmlx:outputVar "country_uri";
  rmlx:function [
    http:methodName "POST";
    http:requestURI "http://localhost/mashup/
      rmlx/country-reconcile{cname}" ];
  rmlx:parameterMapping [rmlx:varName "cname";
  rml:reference "Country" ]];
```

HTTP operations in the web function introduces latency for each call. In most cases, there might be redundancies in data values which make repeating HTTP call unnecessary. By wrapping a remote service endpoint to a local proxy, caching mechanism can be implemented. Thus, network overhead can be reduced.

Further, we implement a new processor capable of transforming new data sources (XLS, HTML, and plain text) which is not supported in the original RML processor. This new data source is expressed as new IRI associated with `rml:referenceFormulation` property.

C. Graphical User Interface

The RMLx front-end (RMLx Editor) is responsible for generating RML mappings while the back-end is actually an RML processor deployed as a web service. The interface for each mapping is a direct form-based translation of each *TripleMapping* instance. Each mapping is described using cards providing general information about the entity that is being mapped. Details like data source, data transformation operation, and additional predicate-object mappings are also hidden using metaphors like collapsibles and reveal. As can be seen in the left rounded rectangle of Figure 2, a mapping card contains generic information such as data source, type of entity, and action buttons. To define new mapping, Generic as well as pefic data source format shortcuts are available from the circular buttons.

IV. EVALUATION

RMLx has been used in several scenarios, of which four are presented in the following subsections. In Table I we show how we use the extended vocabulary and new built-in features in the use-cases. The measured average execution time is presented in Table II.

The measurement is done with a laptop running PHP version 5.6.3 on Windows 8.1 OS with 8GB RAM and intel i7-4500U @1.8GHz. The measurement is taken 11 times and discarding the first measurement. The coverage

⁶<http://www.w3.org/2011/http#>

Fig. 2: Graphical user interface of RMLx (cropped screenshot)

of the measurements includes loading the RML mapping description, loading each data sources, generating the RDF graph, and RDF serialization to Turtle format.

TABLE I: Use-case and RMLx features

use-case (dataset)	ref. binding	data transform	mashup integration
Linked Streaming Widget (lsd)	✓	-	-
Statistical data Integration (wb)	✓	split	✓
Program Committee (pc)	-	disambiguation (user-defined)	-
Open Data Portal (bdg)	✓	string	✓

TABLE II: Measurements of execution time

use-case	data size	format	execution time (s)
lsd	42KB	XML	10.2
wb	10KB	JSON	1.69
pc	340KB	CSV	123.5
bdg	39KB	XML	2.6

A. Linked Streaming Widgets

In the case of *Linked Streaming Widgets* [4], which represent an extension of the Linked Widgets Platform [5], we use RMLx to create streaming RDF data. The lack of ready to use RDF streams on the web created the need to have a means for generating data which can be used as streaming data to feed RDF Stream Processing engines. To this end, we define a

processing pipeline where RMLx mappings are used to convert arbitrary data sources to timestamped RDF data.

RMLx facilitates this process in several ways: First, the appealing arrangement of the user interface supports composition of complex RML data structures. Often the structure of the desired data model reaches a complexity which is difficult to manage without visual aid. Second, the tool allows users to parameterize the conversion process in that variable-like structures can be introduced at runtime. By doing so, for instance, timestamps denoting the observation time of measurements, are added to the data at the time of transformation. This is an essential feature especially in streaming environments. Finally, by providing a RESTful interface, any application is able to incorporate RMLx into its architecture without having to deal with compatibility or platform-dependent issues.

B. Statistical Data Integration

Statistical data is one of the data types which is often published as open data due to its relatively simple tabular-like structure. The diversity of the data's content opens opportunities for creating new analyses and insights if data is integrated from multiple data sources which can be realized via defining RML mappings [6][7].

To demonstrate the use of RMLx, we integrate population data of the United Kingdom from three data sources. These sources are not only different in access methods, but also differ in the published data formats varying between XLS, JSON, XML, and RDF.

Fig. 3: An example of RML mapping (a) and the resulted widget (b)

To overcome the diversity and inconsistency in the published data, we use our extensions of the RML language. Namely, we make use of a new processor which is capable of transforming XLS and XLSX data based on spreadsheet and cell range declarations. Further, we allow users to declare and use variables in the mappings. This allows us to reuse mappings for multiple data sets sharing a similar structure by changing variable values at runtime.

C. Linked Program Committee Data

In this project⁷, RML is used to generate RDF from CSV data that contains Program committee data extracted from conference websites. The use of mapping for RDF generation enables evolution of data model during the project. Changes of data model triggers generation of new RDF data before further checking. Using this method, it can also provide provenance information since the mapping is also in the form of RDF and can be published as Linked Data.

During the generation of RDF, there is a need to disambiguate the entities (affiliation and countries) and link them with appropriate resources from DBpedia. We build REST-based web service to wrap DBpedia lookup service for this task. The service invocation then specified in the extended version of the mapping description.

D. Open Data Mashup

The rise of open data movement worldwide alongside open government movement makes a lot of data now available in open data portals managed by governmental or private institutions. However, some government institutions publish data in form of partial views or slices. This cause the large number of dataset or data resources in the open data portals although some dataset or resources belongs to the same source

of data. This phenomena causes challenges for data retrieval and analysis. End users must do post-processing to integrate these data for their use.

Using RML allows for multiple data source to be included in the same mapping. In Addition, the mapping of RDF data can be exported as data widget in LinkedWidget platform[3][5] allowing for additional exploration and visualization of the data. An example RML mapping generated that is generated from RMLx and the exported widget is shown in Figure 3. In this example, we show the location of public service vehicles in Bandung, Indonesia. We create an RML mapping⁸ of the source data in Figure 3a, generate a widget from it and linked it to the map visualization widget⁹ shown in Figure 3b.

V. RELATED WORK

LOD GRefine [8] is an easy to use extension of Google Refine, which is used to transform tabular data to Linked Data. It is tightly-coupled into Google Refine which is centered around preprocessing task. RDF mapping feature is built on top of this editing environment and the mapping description cannot be exported as Linked Data. Moreover, it only supports tabular data and single data sets to be mapped at a time.

Karma [9] is a data integration system which includes mapping tool to produce RDF from existing data. The interface of mapping tool in Karma is tabular with overlaid visualization of tree which denotes mapping description. The major contribution of Karma is the semi-automatic way of mapping by learn data transformation by example and recommends the mapping based on selected ontology.

The RML Editor [10][11] is a visual interface to create RML mappings originating from the same group who proposed the RML specification. The idea is to let users create mappings

⁷<http://lod.springer.com/wiki/bin/view/Linked+Open+Data/About>

⁸<http://pebbie.org/mashup/rml-source/bandung-gps>

⁹<http://pebbie.org/mashup/widget/map>

based on manipulating a visual representation of the expected output graph. RML Editor approach allows flexible modeling paradigm by providing interfaces for modeling from existing data or target vocabulary.

Our approach focuses on providing an comprehensive environment which comprises of interface for mapping description, graphical RDF visualization for source and produced data, and integration into data exploration environment. Compared to previously mentioned works, our work focuses on tight integration of data exploration pipeline. Using our work as framework, the advantages of these other approach can be adopted to enhance user experience and versatility in preprocessing task.

VI. CONCLUSION

A visual web-based interface, RMLx, for helping users in creating RDF mapping descriptions based on RML has been presented. This interface is built upon the extension of RML vocabulary to address some challenges. We extend RML allowing for (i) the creation of mappings based on new data sources, (ii) introduction of explicit and implicit reference name binding which are replaced at processing runtime, and (iii) the ability to enable data transformations to be included in the mapping description. The feasibility of RMLx is demonstrated by discussing its use in four uses cases dealing with streaming data, statistical data integration, program committee data, and open government data. The reported use-case also shown the practicality of the presented system for real world scenario. Further studies include usability testing compared to other interface, development of semi-automatic mapping generation support, and the feasibility linked data portal based on mapping to enable dynamic linked data generation. Finally, we expect wider adoption of linked data using the presented system.

REFERENCES

- [1] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle, "RML: A generic language for integrated rdf mappings of heterogeneous data," in *LDOW*, 2014.
- [2] F. Michel, L. Djimenou, C. Faron-Zucker, and J. Montagnat, "xR2RML: Relational and non-relational databases to rdf mapping language," Research report. ISRN I3S/RR 2014-04-FR v3, Tech. Rep., 2014.
- [3] T.-D. Trinh, B.-L. Do, P. Wetz, A. Anjomshoa, and A. M. Tjoa, "Linked widgets: An approach to exploit open government data," in *iiWAS*. ACM, 2013, p. 438.
- [4] A. M. Tjoa, P. Wetz, E. Kiesling, T.-D. Trinh, and B.-L. Do, "Integrating streaming data into semantic mashups," *Procedia Computer Science*, vol. 72, pp. 1–4, 2015.
- [5] T. Trinh, P. Wetz, B. Do, A. Anjomshoa, E. Kiesling, and A. M. Tjoa, "Open linked widgets mashup platform," in *AI Mashup Challenge @ ESWC*, 2014.
- [6] B.-L. Do, T.-D. Trinh, P. R. Aryan, P. Wetz, E. Kiesling, and A. M. Tjoa, "Toward a statistical data integration environment: the role of semantic metadata," in *Proceedings of the 11th International Conference on Semantic Systems*. ACM, 2015, pp. 25–32.
- [7] B.-L. Do, P. R. Aryan, T.-D. Trinh, P. Wetz, E. Kiesling, and A. M. Tjoa, "Toward a framework for statistical data integration," *Proceedings of the 3rd International Workshop on Semantic Statistics co-located with 14th International Semantic Web Conference (ISWC 2015)*, 2015.
- [8] M. Verlic, "LODGrefine - LOD-enabled google refine in action," in *I-SEMANTICS 2012 Posters & Demonstrations Track*, 2012, pp. 31–37.

- [9] C. A. Knoblock, P. Szekely, J. L. Ambite, S. Gupta, A. Goel, M. Muslea, K. Lerman, M. Taheriyani, and P. Mallick, "Semi-automatically mapping structured sources into the semantic web," in *Proceedings of the Extended Semantic Web Conference*, Crete, Greece, 2012.
- [10] P. Heyvaert, A. Dimou, A.-L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens, and R. Van de Walle, "RMLEditor: a graph-based mapping editor for Linked Data mappings," in *The Semantic Web – Latest Advances and New Domains (ESWC 2016)*, ser. Lecture Notes in Computer Science, vol. 9678. Springer, May 2016, pp. 709–723. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-34129-3_43
- [11] P. Heyvaert, A. Dimou, R. Verborgh, E. Mannens, and R. Van de Walle, "Towards a uniform user interface for editing mapping definitions," in *Proceedings of the 4th Workshop on Intelligent Exploration of Semantic Data*, 2015.