Automated Metadata Generation for Linked Data Generation and Publishing Workflows

Anastasia Dimou
anastasia.dimou@ugent.be

Tom De Nies
tom.denies@ugent.be

Ruben Verborgh
ruben.verborgh@ugent.be

Erik Mannens
erik.mannens@ugent.be

Rik Van de Walle
rik.vandewalle@ugent.be

Ghent University – iMinds – Data Science Lab

ABSTRACT

Provenance and other *metadata* are essential for determining ownership and trust. Nevertheless, no systematic approaches were introduced so far in the Linked Data publishing workflow to capture them. Defining such metadata remained independent of the RDF data generation and publishing. In most cases, metadata is manually defined by the data publishers (*person-agents*), rather than produced by the involved applications (*software-agents*). Moreover, the generated RDF data and the published one are considered to be one and the same, which is not always the case, leading to pure, condense and often seductive information. This paper introduces an approach that relies on declarative descriptions of (i) mapping rules, specifying how the RDF data is generated, and of (ii) raw data access interfaces to automatically and incrementally generate provenance and metadata information. This way, it is assured that the metadata information is accurate, consistent and complete.

1. INTRODUCTION

Nowadays, data owners publish their data at an increasing rate. More and more of them publish also its corresponding RDF representation and interlink it with other data. However, even though provenance and other metadata become increasingly important, most RDF datasets published in the Linked Data cloud provide no or seldom narrow metadata. To be more precise, only 37% of the published RDF dataset provide provenance information or any other metadata [22]. In these rare cases that such metadata is available, it is only manually defined by the data publishers (*person-agents*), rather than produced by the applications (*software-agents*) involved in the Linked Data publishing cycle. Most of the current solutions which generate and/or publish RDF data, do not consider also automatically generating the corresponding metadata information, despite the well-defined and W3C recommended vocabularies, e.g., PROV-O [16] or VoID [1], that clearly specify the expected metadata output.

As a consequence, the lack of available metadata information neither allow being aware of the origin of the RDF data, nor reproducing the RDF data generation outside the context of the application that originally generated it. This occurs because most of the tools that generate RDF data derived from heterogeneous data, put the focus on independently providing the corresponding RDF representation, dissociating the resulting RDF data from its original source. In the same context, provenance and metadata information regarding the actual mapping rules which specify how the RDF data is generated from raw data, are not captured at all. Nevertheless, such information might equally influence the assessment of the generated RDF data trustworthiness.

Similarly, data publishing infrastructures, such as triple stores, do not automatically publish any provenance or other metadata regarding the RDF data they host. Instead they would have been expected to enrich the metadata produced while the RDF data was generated with metadata associated with the publishing activity. Moreover, the RDF data generation and its publication are considered as interrelated activities that occur together. Although, this is not always the case. Therefore, the generated RDF data and the one subsequently published are not always one and the same. For instance, RDF data might be generated in subsets and published all together, or generated as a single dataset but published in different RDF graphs. Consequently, their provenance and rest metadata information is not identical.

In a nutshell, capturing provenance and metadata information on every step of the Linked Data publishing workflow is not addressed in a systematic and incremental way so far. In this paper, we introduce an approach that considers declarative and machine-interpretable data descriptions and mapping rules to automatically assert provenance as well as other metadata information. Our proposed solution is indicatively applied on mappings described using the RML language [8] and is implemented in the RML tool chain.

The remainder of the paper is structured as follows: In Section 2, we outline the current state of the art. In Section 3, we discuss the essential steps of the Linked Data publishing cycle where provenance and metadata can be generated and in Section 4, we discuss the different levels of metadata details identified. In Section 5, we describe how machine-interpretable mapping rules are considered to automate the metadata generation and in Section 6 we showcase how we implemented it in the RML tool chain.

WWW2016 Workshop: Linked Data on the Web (LDOW2016)

2. STATE OF THE ART

In this section, we investigate existing systems, involved in the Linked Data publishing workflow. Tools generating mappings and RDF data or publish RDF data are approached with respect to their support for automated metadata generation (Section 2.1). In addition, we outline the W3C recommended vocabularies for metadata description (Section 2.2), as well as the most well-known and broadly used approach for representing provenance and other metadata (Section 2.3).

2.1 Linked Data publishing cycle

In the Linked Data publishing workflow there are different activities taking place. Among them, the definition of the rules to generate RDF data from raw data, its actual generation, its publishing and its interlinking are few of the most essential steps. However, the majority of the tools developed to address these tasks do not generate automatically any provenance or metadata information as the corresponding tasks are accomplished, let alone enriching metadata defined in prior steps of the Linked Data publishing workflow.

Hartig and Zhao [10] argued regarding the need of integrating provenance information publication in the Linked Data publishing workflow. However, they focused only on its last step, namely the RDF data publication, outlining metadata publication approaches and showcasing on well-know RDF data publishing tools, such as Pubby[1] and Triplify[2].

None of the well-know systems that generate RDF representations from any type of (semi-)structured data provide any provenance or metadata information in conjunction with the generated RDF data, to the best of our knowledge. For instance, none of the prevalent tools for generating RDF data, such as DB2triples[3], Karma[4], or XSPARQL[5], to indicatively mention a few of the prevalent tools. The main obstacle, at least with respect to provenance, is that it is hard to specify where the data originally resides. That occurs because most of these tools, consider a file as data input. However, where the data of this file is derived from is not known and, therefore, the corresponding provenance annotations can not be accurately defined in an automated fashion.

The D2R server[6] and the CSV2RDF4LOD[7] are the only tool that generates provenance and metadata information in conjunction with the RDF data. However, the D2R server refers only to data in relational databases, it supports a custom provenance vocabulary, not the W3C-recommended PROV-O [16], and is limited to dataset high level metadata information. The CSV2RDF4LOD refers only to CSV files and it achieves capturing provenance using custom bash scripts that aim to keep track of the commands used. The situation aggravates in the case of custom solutions for generating RDF data which neglect to include in its development cycle mechanisms to generate provenance and metadata information.

With the advent of mapping languages, such as the D2RQ[8], SML[9], or the W3C recommended R2RML [4], the mapping rules that specify how triples are generated from raw data, were decoupled from the source code of the corresponding tools that execute them. However, mapping languages are explicitly focused on specifying the mapping rules, neglecting to provide the means to specify the data source too. Whereas, for instance the D2RQ language allows to specify the relational database where the data is derived from, other languages, including R2RML, do not, considering it out of the language's scope. RML [8] is the only language that allows referring to data descriptions based on well-known vocabularies to determine the data source [9] (see Section 5).

The situation remains the same also in the case of interlinking tools, such as the prevalent Silk [28] and Limes [19]. Interlinking tools generate RDF data consisting of links between RDF datasets, the so-called linksets. None of the most well-known tools generate any provenance or metadata annotations regarding the links that were identified and represented as the output dataset of the interlinking task.

In the same context, tools were developed to support data owners to semantically annotate their data. However, those tools still generate both the mapping rules and the corresponding RDF data after the rules execution, without providing any provenance or metadata information. To be more precise, none of the tools that automatically generate mappings of relational databases to its RDF representation, such as BootOx [14], IncMap [21], or Mirror [5], or support users in defining mapping rules, e.g., FluidOps editor [23], supports automated provenance and metadata information generation, neither for the mapping rules, nor for the generated RDF data. Specifying metadata for the mapping rules or considering the mapping rules to determine the provenance and metadata becomes even more cumbersome, in particular in the case of mapping language whose representation is not in RDF, e.g., SML, SPARQL or XQuery.

Similarly, among the RDF data publishing infrastructures, only Triple Pattern Fragments[10] (TPF) [26, 27] provide some metadata information, mainly regarding dataset level statistics and access. Virtuso [11], 4store [12] and other pioneer publishing infrastructures do not provide *out-of-the-box* metadata information, e.g., provenance, dataset-level statics etc. of the RDF data published. LODLaundromat[13] is the only Linked Data publishing infrastructure that provides automatically generated metadata information. However, it uses its own custom ontology[14] which partially relies on the PROV-O ontology to provide metadata information.

2.2 Provenance and Metadata Vocabularies

W3C recommended vocabularies were already defined to specify RDF data provenance and metadata information:

2.2.1 PROV Ontology

The PROV ontology (PROV-O) [16] is recommended by W3C to express the PROV Data Model [18] using the OWL2 Web Ontology Language (OWL2) [13]. PROV-O can be used to represent provenance information generated in different systems and under different contexts.

According to the PROV ontology, a *prov:Entity* is a physical, digital, conceptual, or other kind of thing. A *prov:Activity* occurs over a period of time and acts upon or with entities;

[1] http://wifo5-03.informatik.uni-mannheim.de/pubby/
[2] http://triplify.org/
[3] https://github.com/antidot/db2triples
[4] http://usc-isi-i2.github.io/karma/
[5] http://xsparql.deri.org/
[6] http://d2rq.org/
[7] https://github.com/timrdf/csv2rdf4lod-automation/wiki
[8] http://d2rq.org/d2rq-language
[9] http://sml.aksw.org/
[10] http://linkeddatafragments.org/
[11] http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/
[12] http://4store.org/
[13] http://lodlaundromat.org/
[14] http://lodlaundromat.org/ontology/

it may include consuming, processing, transforming, modifying, relocating, using, or generating entities. A *prov:Agent* bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.

2.2.2 VoID Vocabulary

The Vocabulary of Interlinked Datasets (VOID) [1] is a vocabulary for expressing metadata about RDF datasets with applications ranging from data discovery to cataloging and archiving of datasets. VOID expresses (i) *general*, (ii) *access* and (iii) *structural* metadata, as well as links between datasets. *General metadata* is based on Dublin Core. *Access metadata* describes how the RDF data can be accessed using different protocols. *Structural metadata* describes the structure and schema of the RDF data.

According to the VOID vocabulary, a *void:Dataset* is a set of RDF triples maintained or aggregated by a single provider. A *void:Dataset* is a meaningful collection of triples, that deal with a certain topic, originate from a certain source or process, and contains sufficient number of triples that there is benefit in providing a concise summary. The concrete triples contained in a *void:Dataset* is established through access information, such as the address of a SPARQL endpoint. Last, a *void:Linkset* is a collection of RDF links whose subject and object are described in different datasets.

2.2.3 DCAT Vocabulary

The Data Catalog Vocabulary (DCAT) [17] is designed to facilitate interoperability between data catalogs published on the Web. It aims to (i) increase data discoverability, (ii) enable applications to easily consume metadata from multiple catalogs, (iii) enable decentralized catalogs publishing, and (iv) facilitate federated dataset search.

According to the DCAT vocabulary, a *dcat:Catalog* represents a dataset catalog, a *dcat:Dataset* represents a dataset in the catalog, whereas a *dcat:Distribution* represents an accessible form of a dataset, e.g., a downloadable file, an RSS feed or a Web service that provides the data. DCAT considers as a dataset a collection of data, published or curated by a single agent, and available for access or download in one or more formats. This data is considered for generating an RDF dataset. Thus, the generated RDF dataset forms a *dcat:Distribution* of a certain *dcat:Dataset*.

2.3 Approaches for tracing PROV & metadata

We outline methods for capturing provenance and other metadata information. We identify two approaches that capture provenance and other metadata information inline with the rest RDF data –*Explicit Graphs* (Section 2.3.3) and *Singleton Properties* (Section 2.3.2)– and two that trace them independently of the RDF data –RDF *Reification* (Section 2.3.1) and *Implicit Graphs*(Section 2.3.4). In the following subsections, we discuss in more details alternative approaches for defining the provenance of the following RDF triple:

```
1  ex:item10245 ex:weight "2.4"^^xsd:decimal .
```

2.3.1 RDF Reification

The RDF framework considers a vocabulary for describing RDF statements and providing additional information. RDF reification is intended for expressing properties such as dates of composition and source information, applied to specific instances of triples. The conventional use involves describing an RDF triple using four statements. A description of a statement is called a *reification* of the statement. The RDF reification vocabulary consists of the type `rdf:Statement`, and the properties `rdf:subject`, `rdf:predicate` and `rdf:object`. RDF reification is the W3C recommended approach for representing provenance and metadata information.

```
1  _:ex12345   rdf:type         rdf:Statement .
2  _:ex12345   rdf:subject      ex:item10245 .
3  _:ex12345   rdf:predicate    ex:weight .
4  _:ex12345   rdf:object       "2.4"^^xsd:decimal .
5  _:ex12345   prov:wasDerivedFrom _:src123 .
```

The major disadvantage of RDF reification is the number of triples required to represent a reified statement. For each generated triple, at least four additional statements is required to be generated. So, for an RDF dataset of N triples, the metadata graph will be equal to four times the number of the RDF dataset triples in the best case where only the RDF reification statements are generated and no additional.

2.3.2 Singleton Properties

Singleton properties [20] is an alternative approach for representing statements about statements using RDF. This approach relies on the intuition that the nature of every relationship is universally unique and can be a key for any statement using a singleton property. A *singleton property* represents one specific relationship between two entities under a certain context. It is assigned a URI, as any other property, and can be considered as a subproperty or an instance of a generic property. Singleton properties and their generic property are associated with each other using the `singletonPropertyOf` property, subproperty of `rdf:type`.

```
1  ex:item10245   ex:weight#1              "2.4"^^xsd:decimal .
2  ex:weight#1    sp:singletonPropertyOf   ex:weigh .
3  ex:weight#1    prov:wasDerivedFrom      _:src123 .
```

2.3.3 Explicit Graphs

The *Explicit Graphs* approach relies on *named graphs*. *Named Graphs* is a set of RDF triples named by a URI and can be represented using *TriG* [3], *N-Quads* [2] or *JSON-LD* [24], but it is not compatible with all RDF serialisations. This approach is similar to *Singleton Properties*. Instead of annotating the common predicate of the triples, the context of the triple is annotated. This way, introducing one triple per predicate is avoided. However, the *Explicit Graphs* approach has two drawbacks: (i) they are not supported by all RDF serializations; and (ii) they might be in conflict with the named graph defined as part of the RDF dataset and whose intent is different than tracing provenance information.

```
1  ex:item10245   ex:weight  "2.4"^^xsd:decimal    ex:graph .
2  ex:graph       prov:wasDerivedFrom    _:src123 .
```

2.3.4 Implicit Graphs

Implicit graphs are URIs assigned implicitly to a dataset, graph, triple or term. An *Implicit Graph* is aware of what it represents but the represented entity is not directly linked to its implicit graph. Implicit graphs might be used to identify a dataset or a graph, but also triples. In the later case, as Triple Pattern Fragments (TPF) introduced [26, 27], each triple can be found by using the elements of itself, thus, each triple has a URI and, thereby, its implicit graph. For example, the triple `x y z` for a certain dataset could be identified by the TPF URI http://example.org/dataset?subject=x&predicate=y&object=z.

```
1  <http://example.org/dataset?
2  subject=ex:item10245&predicate=ex:weight&object="2.4">
3      prov:wasDerivedFrom _:src123 .
```

3. WORKFLOW METADATA STEPS

Provenance and other metadata information can be captured at different steps of the publishing workflow. Keeping track of metadata derived from the different steps of the RDF data generation and publishing workflow, results in more complete information regarding how an RDF dataset was generated and formed in the end. Moreover, provenance and metadata information generated at different steps of the publishing workflow offer complementary information.

We identify the following primary steps: mapping definitions generation (Section 3.1), data source retrieval (Section 3.2), RDF data generation (Section 3.3), RDF data publication (Section 3.4). We consider each workflow step as an activity (`prov:Activity`) whose properties is needed to be traced. In Table 1, we summarize those activities and the information that needs to be defined each time. The provenance and how the different steps are associated with each other are shown at Figure 1.

		Same Dataset		Different Dataset		
	Map.	**Gen.**	**Pub.**	**Gen.**	**Pub.**	**Link.**
prov:Entity						
prov:wasGeneratedBy	●	●		●	●	●
prov:wasDerivedFrom	●	●		●	●	●
prov:wasAttributedTo		●	●	●	●	●
prov:Agent	●	◐	◐	●	●	●
prov:actedOnBehalfOf	●	●	●	●	●	●
void:Dataset − General						
dcterms:creator	●	●		●		●
dcterms:contributor	●	●	●	●	●	●
dcterms:publisher			●		●	
dcterms:source		●			●	
dcterms:created	●	●		●	●	
dcterms:modified	●	●		●	●	
dcterms:issued	●	◐	◐	●	●	
dcterms:license	●	◐	◐	●	●	●
void:feature	●	◐	◐	●	●	●
void:Dataset − Access			●			●
void:Dataset − Structural	●	◐	◐	●	●	●
void:Dataset − Statistics	●	◐	◐	●	●	●
void:Linkset	●	◐	◐	●	●	●

A filled circle (●) indicates that the property should be (re-)assessed in *each* of the marked steps. A half-filled circle (◐) indicates that that property can be assessed in *any* of the marked steps.

Table 1: Table of properties required for each entity

3.1 Mapping Rules Definition

Provenance and metadata information is required to be captured when the mapping rules are defined (Fig. 1, Edit Map Doc). In this case, it is important to track *when* the mapping rules were edited or modified and by *whom*. An RDF dataset might have been generated using multiple mapping rules whose definition occurred at different moments and by different agents. Consequently, the generation of certain mapping rules (Fig. 1, *Generate Map Doc*) is an activity (`prov:Activity`) which is informed by all prior editing activities (Fig. 1, *Edit Map Doc*). For instance, a mapping rule might have been generated by a mapping generator or edited by a human-agent using a mapping editor. However, such a maping rule might have been modified or used in conjunction with other mapping rules which were generated, in their own turn, by another human-agent at a different time.

The agent who defined the mapping rules (Fig. 1, *Mapping Editor*) might differ from the one who generated (Fig. 1, *Data Generator*) or published the data (Fig. 1, *Data Publisher*), or even the owner of the data (Fig. 1, *Data Owner*). Being aware of who defined the mapping rules is of crucial importance to assess the trustworthiness of the final RDF data, even though it is neglected so far. For instance, RDF data generated using mapping rules from an automated generator might be considered less trustworthy compared to RDF data whose mapping rules were defined by a data specialist.

3.2 Data Sources Retrieval

An RDF dataset might be derived from one or more heterogeneous data sources (Fig. 1, *Data Source Acquisition*). Each data source, in its own turn, might be derived from an input. For instance, a table might be derived from a database or some JSON data might be derived from a Web API. Such a data source might be turned into an RDF graph partially or in its entirety. This might mean that not the entire stored data is retrieved but a selection is only used to generate the RDF data. For instance, only the data that fulfils an SQL query could be retrieved to generate the RDF dataset, instead of the entire table or database.

For this activity, it is important to keep track of metadata regarding the data sources and their retrieval, as this indicates the original data sources of the generated RDF data. However, the originally stored data might have changed over time. For instance, in the case of an API, some data is retrieved at a certain time, but different data might be retrieved at a subsequent time. Therefore, it is crucial to know when the data is accessed to assess its timeliness with the original data. For instance, comparing the last modified date of the original data and the generation date of the RDF data, indicates whether the available RDF representation is aligned with the current version of the original data or not.

3.3 RDF Data Generation

As soon as the mapping rules and the data source are available, the RDF data is generated (Fig. 1, *Generate RDF Data*). For this activity, it is important to keep track of (i) *how* the RDF data generation was triggered, i.e. *data-driven* or *mapping-driven*, from raw data (RDF generation) or from RDF data (RDF interlinking); (ii) *when* the RDF dataset was generated, and (iii) *how*, i.e. in a single dataset or in subgraphs, subsets etc. Besides the aforementioned, this activity is crucial for capturing the origin of the RDF data, as only at this step that information is known (in combination with the data description and acquisition).

3.4 RDF Data Publication

The published RDF data is not always identical to the generated one (Fig. 1, *genRDF Vs. pubRDF*). For instance, it might be the result of merging multiple RDF datasets which are generated from different data sources at the same or different moments. Moreover, the published RDF dataset might be published in a different way compared to how the RDF data was generated. For instance, it could be split in different graphs to facilitate its consumption. This might lead to

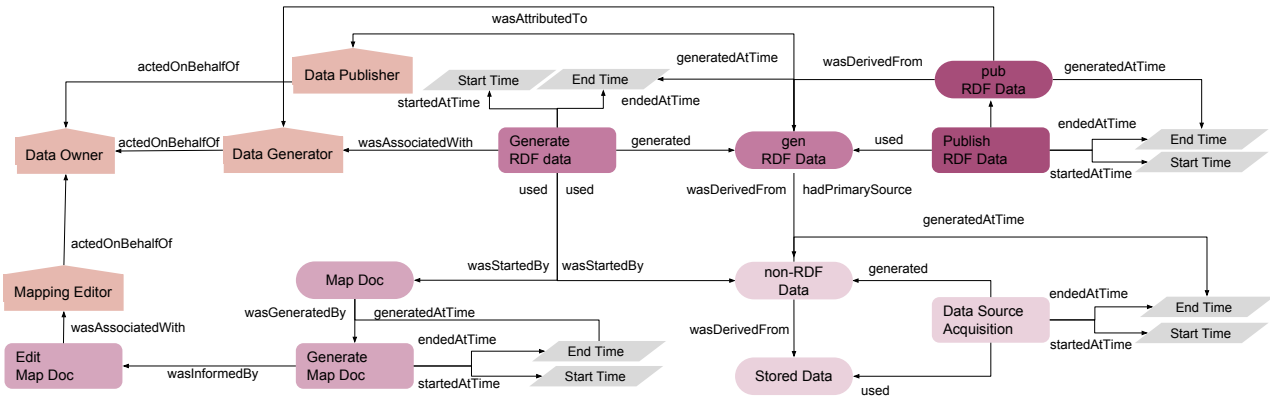

Figure 1: A coordinated view of Linked Data publishing workflow activities.

different metadata for the generation and publication activities, and these metadata sets might have different purposes.

For instance, VoID access information metadata is more meaningful and possible to be generated during the RDF data publication, whereas provenance information in respect to the original data can only be defined during the RDF data generation activity. To the contrary, VoID structural or statistical metadata might be generated both during RDF data generation and publication. However, the generated RDF data is not always identical to the one published. If the generated RDF data differ from the one published, then such metadata should be defined for both cases (see Table 1).

4. METADATA DETAILS LEVELS

There are different details levels for capturing provenance and metadata information. However, in most cases so far, the provenance and metadata information is delivered on dataset level. This mainly occurs because the metadata information are only defined after the RDF data is generated and/or published. However, different applications and data consumption cases require different levels of provenance and metadata information. Overall, the goal is to achieve the best trade-off between details level and number of additional triples generated for balancing information overhead and acceptable information loss in an automated metadata generation occasion. For instance, considering RDF reification for capturing all provenance and metadata information for each triple, means that metadata referring to the entire RDF dataset is captured repeatedly for each individual triple. To the contrary, considering an implicit graph on dataset level results in information loss in respect to the origin of each triple, if multiple data sources are used to generate the RDF dataset, because it is not explicitly defined where each triple is derived from.

Automating the provenance and metadata information generation, allows exploiting hybridic approaches which can contribute in optimizing the metadata information balance. In this section, we outline the different details levels for capturing metadata that we identified: *Dataset level* (Section 4.1), *named graph level* (Section 4.2), *partition level* (Section 4.3), *triple level* (Section 4.4) and *term level* (Section 4.5). For each level, we describe what type of metadata is captured and we discuss the advantages and disadvantages when used in combination with different representation approach.

4.1 Dataset Level

Dataset level provenance and metadata provide high-level information for the complete RDF dataset. This level of detail is meaningful for all metadata information that refer to the whole dataset, i.e. a `void:Dataset` and are the same for each triple. Therefore, among the alternative representation approaches, considering an *explicit* or *implicit graph* for the dataset to represent provenance and metadata annotations is sufficient on *dataset level* and it requires the least number of additional triples. The alternative approaches in principle assign the same metadata information to each triple. Thus, the exact same information is replicated for each triple, causing unnecessary overhead.

Provenance information on *dataset level* is sufficient if all triples are derived from the same original data source and are generated at the same time, as a result of a single activity. The same occurs if the overall origin source is sufficient to assess the RDF dataset trustworthiness. On the contrary, if being aware of the exact data source is required, for instance to align the semantically annotated representation with the original data values, more detailed provenance information is desired, because the high level provenance information is not as complete and accurate to accomplish the desired task.

4.2 Named Graph Level

An RDF dataset might consist of one or more named graphs. Named graph based subsets of an RDF dataset provide conceptual partitions of RDF triples semanticfully distinguished in graphs. *Named graph level* provenance and metadata information refer to all RDF annotations which are related to a certain named graph and contain information for each one of the named graphs. Each named graph is a `void:Dataset` and consists a subset of the whole RDF dataset.

In the case of *named graphs*, it is not possible to represent metadata and provenance information using explicit graphs, because the RDF statements are already *quads* and the named graph has different semantics than providing metadata information. As in the case of *dataset level*, *implicit graphs* for each named graph and for the complete dataset generate the minimum number of additional RDF triples. Moreover, the *named graph level* metadata information are sufficient if all triples of a certain named graph are derived from the same data source. Otherwise, there is information loss which can be addressed at a narrower detail level.

4.3 Partitioned Dataset Metadata Level

A dataset might be partitioned based on different aspects. The most frequent partitions are related to (i) the underlying data source or the triple's (ii) subject, (iii) predicate, or (iv) object. Besides the aforementioned partitions, any other custom partition can be equally considered. A *source-based* partitioned RDF dataset is an RDF dataset whose subsets are formed with respect to their derivation source. To be more precise, all RDF terms and triples are derived from the same original data source. *Source-based* partitioned RDF datasets derived from a single data source are not considered because they coincide with the actual RDF dataset. A *subject-based* partitioned RDF dataset is the part of an RDF graph whose triples share the same subject. Consequently, *subject-level* metadata provides information for all triples which share the same subject. It similarly applies in the case of *predicate-based* or *object-based* partitions.

Partitioned datasets might be treated in the same way as named graphs, but it is also possible to use *explicit graphs* to define the subsets metadata. An *implicit graph* for each subset of the RDF dataset which resembles a partition achieves generating the minimum number of additional triples for the metadata information. In the particular case of *predicate-based* partition, representing the provenance and metadata information using *singleton properties* would cause generating almost the same number of additional triples as in the case of defining an *explicit* or *implicit graph* per partition.

4.4 Triple Level

If metadata is captured on *triple level*, it becomes possible to keep track of the data source each triple was derived from. However, that causes the generation of RDF annotations for metadata whose number of triples is larger than the actual dataset. In the simplest case, the number of additional triples for the metadata information depends on the number of data sources. The more data sources, the more metadata information to be defined. *Triple level* metadata become meaningful also in the case of big data or streamed data where the time one triple was generated might significantly differ compared to the rest triples of the RDF dataset.

In the case of *triple level* metadata, singleton properties become meaningful when statements about all triples sharing the same property share the same metadata information. For instance, if all triples whose RDF terms are associated using a certain predicate, share the same metadata, e.g., they are all derived from the same data source.

4.5 RDF Term Level

Even RDF terms that are part of a certain RDF triple can derive from different data sources. For instance, an RDF term is generated considering some data value derived from a source A. This RDF term might constitute the subject of an RDF triple whose object though is an RDF term derived from a source B. In this case, even more detailed metadata information is required to keep track of the provenance information. Among the alternative approaches for representing metadata, the RDF reification becomes meaningful at this level of detail. To be more precise, the RDF reification is meaningful in the cases that the RDF terms that consist an RDF triple and form a statement derive from different data sources and/or are generated at a different time.

5. METADATA GENERATION WITH RML

We introduce an approach that takes into consideration machine interpretable descriptions of data sources and mapping rules, which are used to generate RDF datasets, to also automatically generate its corresponding provenance and metadata information. Our approach relies on asserting statements from declarative descriptions of data sources and mapping rules. This allows our proposed approach to be applied on alternative mapping languages and be replicated in different implementations.

In our exemplary case, machine interpretable mapping rules are defined using the RDF Mapping Language (RML) [8]. RML is considered because it is the only language that allows uniformly defining the mapping rules over heterogeneous data sources. Moreover, RML is aligned with machine interpretable data source descriptions defined using different vocabularies, e.g., DCAT [17], CSVW [25], Hydra [15] etc [9].

5.1 RML Mapping Definitions

Mapping rules are defined using the RDF Mapping Language (RML). RML [8] extends the W3C recommended R2RML mapping language [4] defined for specifying mappings of data in relational databases to the RDF data model. RML covers also mappings from data sources in different (semi-)structured formats, such as CSV, XML, and JSON.

RML documents contain rules defining how the input data can be represented in RDF. An RML document (see Listing 1) contains one or more Triples Maps (line 5 and 13). A Triples Map defines how triples are generated and consists of three main parts: the Logical Source, the Subject Map and zero or more Predicate-Object Maps. The Subject Map (line 6 and 14) defines how unique identifiers (URIs) are generated for the resources and is used as the subject of all RDF triples generated from this Triples Map. A Predicate-Object Map (line 7 and 15) consists of Predicate Maps, which define the rule that generates the triple's predicate (line 9, 17 and 19) and Object Maps (line 18 and 20) or Referencing Object Maps (line 10), which define how the triple's object is generated.

```
1  @prefix rr:     <http://www.w3.org/ns/r2rml#>.
2  @prefix rml:    <http://semweb.mmlab.be/ns/rml#>.
3  @prefix foaf:   <http://xmlns.com/foaf/0.1/>.
4
5  <#PersonMap> rml:logicalSource <#DCAT_LogicalSource> ;
6               rr:subjectMap <#PersonSubjectMap>;
7               rr:predicateObjectMap <#AccountPreObjMap>.
8  <#PersonSubjectMap> rr:template "http://ex.com/{ID}".
9  <#AccountPreObjMap> rr:predicate foaf:account;
10                     rr:objectMap <#TwitterRefObjMap>.
11 <#TwitterRefObjMap> rr:parentTriplesMap <#TwitterAcount>.
12
13 <#TwitterAcountMap> rml:logicalSource <#DB_LogicalSource>;
14                     rr:subjectMap <#TwitterSubMap>;
15 rr:predicateObjectMap <#AccountPreObjMap>, <#HomepagePreObjMap>.
16 <#TwitterSubMap>     rr:template "http://ex.com/{account_ID}".
17 <#AccountPreObjMap>  rr:predicate foaf:accountName;
18                      rr:objectMap [ rml:reference "name"].
19 <#HomepagePreObjMap> rr:predicate foaf:accountServiceHomepage;
20                      rr:objectMap <#HomepageObjMap>.
21 <#HomepageObjMap>    rml:reference "resource".
```

Listing 1: RML Mapping Rules

5.2 Mapping Document Metadata

A mapping document summarizes mapping rules defined using the RML language. RML is serialized in RDF, thus a mapping document (<#MapDoc>) can be considered as an RDF dataset itself (void:Dataset). Therefore, it has its own

metadata as any other RDF data can have. To be more precise, a mapping document is a prov:Entity that can be associated with a prov:Agent, either a human agent or software. The Mapping Document is the result of a prov:Activity, which is informed, on its own turn, from different editing activities.

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix prov:    <http://www.w3.org/ns/prov#>.
@prefix void:    <http://rdfs.org/ns/void#>.

<#MapDoc> a prov:Entity, void:Dataset;
 prov:generatedAtTime   "2016-01-05T17:10:00Z"^^xsd:dateTime;
 prov:wasGeneratedBy    <#MapDoc_Generation>;
 prov:wasAssociatedWith <#RMLEditor>;
 prov:wasAttributedTo <http://rml.io/people/AnastasiaDimou>;
 dcterms:creator      <http://rml.io/people/AnastasiaDimou>;
 dcterms:created      "2016-01-05T17:10:00Z"^^xsd:dateTime;
 dcterms:modified     "2016-01-05T17:15:00Z"^^xsd:dateTime;
 dcterms:issued       "2016-01-07T10:10:00Z"^^xsd:dateTime.

<#MapDoc_Editing>     a prov:Activity;
 prov:startedAtTime   "2016-01-05T17:00:00Z"^^xsd:dateTime;
 prov:endedAtTime     "2016-01-05T17:10:00Z"^^xsd:dateTime .

<#MapDoc_Generation>  a prov:Activity;
 prov:generated       <#MapDoc>;
 prov:startedAtTime   "2016-01-05T17:09:00Z"^^xsd:dateTime;
 prov:endedAtTime     "2016-01-05T17:10:00Z"^^xsd:dateTime;
 prov:wasInformedBy   <#MapDoc_Editing>.

<#RMLEditor> a prov:Agent;
 prov:type prov:SoftwareAgent.

<http://rml.io/people/AnastasiaDimou> a prov:Agent;
 prov:type prov:Person;
 prov:actedOnBehalfOf <#DataOwner>.
```

Listing 2: Mapping Metadata Description

Besides the metadata regarding the entire mapping document (<#MapDoc>), similarly metadata might be defined on Triples Map level or regarding any of the Term Maps, especially in case that different parts of the mapping document (subsets of <#MapDoc>) were defined by different agents or at different times. For instance, the metadata information of different Triples Map might be as follows:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix prov:    <http://www.w3.org/ns/prov#>.
@prefix void:    <http://rdfs.org/ns/void#>.

<#MapDoc> void:subset <#PersonMap>, <#TwitterAccountMap>.

<#PersonMap>            a prov:Entity, void:Dataset;
 prov:wasGeneratedBy    <#PersonMap_Generation>;
 prov:wasAssociatedWith <#RMLEditor>;
 prov:wasAttributedTo   <http://rml.io/people/AnastasiaDimou>.

<#TwitterAccountMap>    a prov:Entity, void:Dataset;
 prov:wasGeneratedBy    <#TwitterAccountMap_Generation>;
 prov:wasAssociatedWith <#RMLEditor>;
 prov:wasAttributedTo   <http://rml.io/people/AnastasiaDimou>.

<#PersonMap_Generation> a prov:Activity;
 prov:generated         <#PersonMap>;
 prov:wasInformedBy     <#PersonMap_Editing>.

<#TwitterAccountMap_Generation> a prov:Activity;
 prov:generated         <#TwitterAccountMap>;
 prov:wasInformedBy     <#TwitterAccountMap_Editing>.

<#PersonMap_Editing>         a prov:Activity.
<#TwitterAccountMap_Editing> a prov:Activity.
```

Listing 3: Triples Map Metadata Description

5.3 Data Sources Retrieval Metadata

One or more data sources might be considered for generating an RDF dataset. In our exemplary case, one data source is described by the <#DB_LogicalSource> and the underlying database that contains the data is described by the <#DB_Source> using the D2R vocabulary. Its description:

```
@prefix rml:  <http://semweb.mmlab.be/ns/rml#>.
@prefix dcat: <http://www.w3.org/ns/dcat#>.
@prefix d2rq:
 <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.

<#DB_LogicalSource> rml:logicalSource [
  rml:query """SELECT * FROM DEPT WHERE ... """ ;
  rml:source <#DB_Source> ].

<#DB_Source>      a d2rq:Database;
  d2rq:jdbcDSN    "jdbc:mysql://localhost/example";
  d2rq:jdbcDriver "com.mysql.jdbc.Driver";
  d2rq:username   "user";
  d2rq:password   "password".
```

Listing 4: Database Source description

Similarly, a data source might be a *dcat:Dataset* and one of its distributions might be considered for generating the RDF dataset. Directly downloadable distributions contain a *dcat:downloadURL* reference. For instance:

```
@prefix rml:  <http://semweb.mmlab.be/ns/rml#>.
@prefix dcat: <http://www.w3.org/ns/dcat#>.

<#DCAT_LogicalSource> rml:source <#DCAT_Source>;
                      rml:referenceFormulation ql:XPath;
                      rml:iterator "...".

<#DCAT_Source> a dcat:Dataset;
               dcat:distribution <#XML_Distribution> .

<#XML_Distribution> a dcat:Distribution;
                    dcat:downloadURL <http://ex.org/file.xml>.
```

Listing 5: DCAT source description

The data source retrieval can be considered as a prov:Activity attributed to a prov:Agent. Such a prov:Agent can be the *data owner* or an agent acting on his behalf, i.e. *Data Generator*. The data source consists of a prov:Entity which was derived from the data acquisition activity. The original data source description provides some information regarding the data source. Additional, provenance information is added for further clarity. The metadata for a data source, e.g., the <#DB_LogicalSource> and the <#DCAT_LogicalSource>, are described as follows:

```
@prefix prov: <http://www.w3.org/ns/prov#>.

<#DB_LogicalSource>   a prov:Entity;
 prov:wasDerivedFrom  <#DB_Source> ;
 prov:generatedAtTime "2016-01-05T17:10:00Z"^^xsd:dateTime .

<#DB_Retrieval>     a prov:Activity;
 prov:generated     <#DB_LogicalSource> ;
 prov:used          <#DB_Source>;
 prov:startedAtTime "2016-01-05T17:00:00Z"^^xsd:dateTime ;
 prov:endedAtTime   "2016-01-05T17:10:00Z"^^xsd:dateTime .

<#DCAT_LogicalSource> a prov:Entity;
 prov:generatedAtTime "2016-01-05T17:05:00Z"^^xsd:dateTime .

<#DCATsource_Retrieval> a prov:Activity;
 prov:generated         <#DCAT_LogicalSource>;
 prov:used              <#DCAT_Source>.
```

Listing 6: Data Sources Metadata Description

5.4 RDF Dataset Generation Metadata

Considering the aforementioned mapping document and the data source descriptions, RDF triples are generated.

Approaches for Tracing Metadata and RML

Among the different approaches for capturing provenance and metadata information, RML can best be aligned with *implicit graphs* and RDF *reification*. Those two approaches generate metadata information independently of the generated RDF data. In particular, *explicit graphs* are not considered as they might coincide with the *named graphs* if any explicitly defined for the actual RDF dataset.

Metadata Details Levels and RML

Dataset level metadata information is associated with all triples generated considering all mapping rules in a mapping document. *Named graph level* metadata information is associated with triples which are generated considering Term Maps related to the corresponding Graph Map.

As far as partitioned RDF datasets is concerned, each partition is associated with different parts of one or more Triples Maps. To be more precise, *source-level* metadata information is generated for all triples which are derived from Triples Maps which share the same Logical Source. In the same context, *subject-level* metadata information is generated for each unique instantiation of one (or more) of the Subject Maps that appear in the mapping document. Similarly, *predicate-level* metadata information is generated for each unique predicate which appears in one or more Triples Maps. Last, *object-level* metadata anotations are generated for each unique object which is generated due to an Object Map. Whereas the aforementioned levels consider *implicit graphs* to represent provenance and metadata information, *triple* and *RDF term level* metadata information can only be captured considering reification statements.

Dataset Level Metadata

Dataset level provenance and metadata information has as follows for the aforementioned running example:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix prov:    <http://www.w3.org/ns/prov#>.

<#RDF_Dataset> a prov:Entity, void:Dataset;
 prov:generatedAtTime   "2016-01-05T17:10:00Z"^^xsd:dateTime;
 prov:wasGeneratedBy    <#RDFdataset_Generation>;
 prov:wasDerivedFrom <#DB_LogicalSource>,<#DCAT_LogicalSource>;
 prov:wasAssociatedWith <#RMLProcessor>;
 prov:wasAttributedTo <http://rml.io/people/AnastasiaDimou>;
 dcterms:creator      <http://rml.io/people/AnastasiaDimou>;
 dcterms:created      "2016-01-05T17:10:00Z"^^xsd:dateTime;
 dcterms:modified     "2016-01-05T17:12:00Z"^^xsd:dateTime;
 dcterms:issued       "2016-01-07T10:10:00Z"^^xsd:dateTime.

<#RDFdataset_Generation> a prov:Activity;
 prov:generated          <#RDF_Dataset>;
 prov:startedAtTime      "2016-01-05T17:00:00Z"^^xsd:dateTime;
 prov:endedAtTime        "2016-01-05T17:10:00Z"^^xsd:dateTime;
 prov:wasInformedBy      <#MapDoc_Generation>;
 prov:used <#MapDoc>,<#DB_LogicalSource>,<#DCAT_LogicalSource>.

 <#RMLProcessor> a prov:Agent;
  prov:type prov:SoftwareAgent.
```

Listing 7: RDF Dataset Level Metadata

The RDF data generation might be triggered either by a mapping document (*mapping-driven approach*) or by a data source (*data-driven approach*) [11]. Depending on which approach occurs at a certain case, the RDF data generation activity (<#RDFdataset_Generation>) is informed by the mapping document generation activity (<#MapDoc_Generation>) or by the data source generation activity (e.g., the <#DB-source_Retrieval> or the <#DCATsource_Retrieval>).

Specifying the data source where the RDF dataset was derived from becomes easy and can be automatically asserted thanks to the aligned mapping and data source descriptions. The RML mapping rules declaratively define the data sources used, in contrast to other mapping languages which do not explicitly define the data sources considered for fulfilling the mapping activity.

However, as one can observe, it is defined that the RDF dataset was derived from an extract of data from a database and an XML file published on the Web, but it is not explicitly defined which triples are derived from each data source.

Triple Level Metadata

In order to address the aforementioned ambiguity regarding the RDF triples origin, RML metadata generation might be defined based on the Predicate Object Maps, for instance the <#AccountPreObjMap> and the <#HomepagePreObjMap>. The metadata information of the generated RDF triples follows:

```
@prefix prov: <http://www.w3.org/ns/prov#>.
@prefix void: <http://rdfs.org/ns/void#>.

<#RDF_Dataset> void:subset <#AccountRDF>, <#HomepageRDF>.

<#AccountRDF> a prov:Entity, void:Dataset;
 prov:wasGeneratedBy    <#RDFdataset_Generation>;
 prov:wasDerivedFrom <#DB_LogicalSource>,<#DCAT_LogicalSource>;
 prov:wasAssociatedWith <#RMLProcessor>;
 prov:wasAttributedTo   <http://rml.io/people/AnastasiaDimou>.

<#HomepageRDF> a prov:Entity, void:Dataset;
 prov:wasGeneratedBy    <#RDFdataset_Generation>;
 prov:wasDerivedFrom    <#DB_LogicalSource>;
 prov:wasAssociatedWith <#RMLProcessor>;
 prov:wasAttributedTo   <http://rml.io/people/AnastasiaDimou>.
```

Listing 8: RDF Triple Level Metadata

Even though it is easy one to observe that this resolves the ambiguity issue regarding the provenance in the case of triples generated considering the <#HomepagePreObjMap>, it is not the same in the case of triples generated considering the <#AccountPreObjMap>. In the later case, the RDF triples are formed generating the subject and the object from different data sources. To be more precise, the triple's subject is generated considering a value derived from the <#DCAT_LogicalSource>, whereas the triple's object is derived from the <#DB_LogicalSource>. If, even more detailed provenance is required, RDF *term level* should be preferred.

RDF Term Level Metadata

The RDF *term level* is the narrowest details level for metadata information. It is applicable in the cases of Referencing Object Maps, namely when the subject and the object of an RDF triple is derived from different data sources. RDF reification is the only approach for representing this metadata information. Considering the aforementioned running example, the metadata information of the RDF triples generated from the <#AccountPreObjMap> Predicate Object Map are defined as it follows:

```
_:ex12345   rdf:type        rdf:Statement .
_:ex12345   rdf:subject     ex:item10245 .
_:ex12345   rdf:predicate   foaf:account .
_:ex12345   rdf:object      <https://twitter.com/natadimou>.

ex:item10245 prov:wasDerivedFrom <#DCAT_LogicalSource>.
<http://twitter.com/natadimou>
 prov:wasDerivedFrom <DB_LogicalSource>.
```

Listing 9: RDF Term Level Metadata

DCAT Catalogue Enrichment

In the case that the original data is published on the Web in the frame of a catalogue described with the DCAT vocabulary, complementary metadata information can be generated to enrich it. If one of the dcat:Dataset distributions (dcat:Distribution) is considered to generate the corresponding RDF representation, complementary DCAT metadata might be generated as well, to specify that the generated RDF dataset is another distribution of a certain dcat:Dataset published on the dcat:Catalog. For instance, in the case of the running example, the <#DCAT_RDF> and the <#DB_RDF> are *source-level* partitions of the RDF dataset. The RDF triples in the <#DCAT_RDF> partition is an RDF distribution of the <#XML_Distribution> for the <#DCAT_Source>. The <#DCAT_RDF> metadata information and the enriched <#DCAT_Source> have as follows:

```
@prefix dcat: <http://www.w3.org/ns/dcat#>.
@prefix prov: <http://www.w3.org/ns/prov#>.
@prefix void: <http://rdfs.org/ns/void#>.

<#RDF_Dataset> void:subset <#DCAT_RDF>, <#DCAT_RDF>.

<#DCAT_RDF> a prov:Entity, void:Dataset;
 prov:wasGeneratedBy    <#RDFdataset_Generation>;
 prov:wasDerivedFrom    <#DCAT_LogicalSource>;
 prov:wasAssociatedWith <#RMLProcessor>;
 prov:wasAttributedTo <http://rml.io/people/AnastasiaDimou>.

<#DCAT_Source> dcat:distribution <#DCAT_RDF>.
```

Listing 10: DCAT Metadata Enrichment

6. METADATA & THE RML TOOL CHAIN

We implemented the aforementioned approach at the RML tool chain, namely the RMLEditor[15] and the RMLProcessor[16]. The RML tool chain was configured to support *implicit graphs* and RDF *reification*. The supported metadata can be further extended to take into consideration other metadata vocabularies too and generate corresponding metadata information. In more details:

RML Editor

The RMLEditor [12] was extended to generate metadata regarding the editing and generation of the mapping rules, as they are declaratively represented using the RML language. The RMLEditor keeps track of the mapping document edition and generation activities, when they occurred and by whom. The RMLEditor was extended to support *implicit graphs* for defining the metadata information which is related to the RML mapping document or its subsets.

[15] http://rml.io/RMLeditor.html

[16] http://github.com/RMLio/RML-Mapper

RML Processor

The RMLProcessor was extended with a Metadata Module that automatically generates metadata for the generated RDF dataset. The desired metadata to be generated by the RMLProcessor can be configured by an agent1. The agent who triggers the mapping activity can define the vocabulary to be used, as well as the desired details level. By default, the W3C recommended PROV [16], VOID [1] and DCAT [17] vocabularies are supported. Although, the RMLProcessor can be further extended to support other vocabularies and generate more metadata information.

The RMLProcessor has also been extended to automatically generate corresponding metadata regarding RDF dataset generation. It was extended to generate metadata information considering implicit graphs or RDF reification. To be more precise, the RMLProcessor was configured to generate metadata information using *implicit graphs* for the metadata which are related with the whole dataset as well as for named graphs. The RMLProcessor was configured to generate RDF reification triples if the metadata level is set on triple or RDF term level. The *explicit graphs* and *singleton properties* were not considered because they need to be defined inline with the actual RDF data, together with the mapping rules.

7. CONCLUSIONS AND FUTURE WORK

The proposed approach aims to show how metadata of the fundamental activities for the generation and publication of RDF triples can be automatically generated. Our solution covers the RDF dataset generation, including metadata for the mapping rules definition and the data descriptions. Based on the provided metadata information, it is expected that publishing infrastructures will enrich this information with complementary details regarding the RDF dataset publication activity. Moreover, it is expected that RDF publication infrastructures will re-determine certain properties regarding the metadata information, if the RDF dataset is reformed before it gets published. Moreover, the metadata can be enriched with additional information derived from other activities involved in the Linked Data publishing workflow.

Provenanve and metadata information can be multidimensional and its consumption diverges across different systems. Different applications require different levels of metadata information to fulfil their tasks, whereas diverse metadata information might be desired. The presented workflow was focused on the essential parts of the Linked Data publishing workflow. However, any metadata information might be considered as they accrue from other activities involved in the Linked Data publishing workflow. In the future, we consider including metadata information regarding the results of the RDF validation [7, 6], applied both on the mapping document, as well as on the generated RDF data.

Different aspects of the RDF data generation and publishing might influence its quality and trustworthiness assessment. In most of the cases so far, the provenance and metadata information are manually delivered on dataset level. Automating the provenance and metadata generation relying on machine interpretable descriptions of the different workflow steps, allows to generate metadata in a systematic way. The generated provenance and metadata information becomes more accurate, consistent and complete. The metadata generation for certain RDF data is an incremental procedure that relies on the contribution of different activ-

ities in the Linked Data publishing workflow to enrich the information we have for the generated RDF dataset.

8. ACKNOWLEDGMENTS

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.

9. REFERENCES

[1] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets with the VoID Vocabulary. W3C Interest Group Note, Mar. 2011. http://www.w3.org/TR/void/ .

[2] G. Carothers. RDF 1.1 N-Quads. Working Group Recommendation, W3C, Feb. 2014. https://www.w3.org/TR/n-quads/ .

[3] G. Carothers and A. Seaborne. RDF 1.1 TriG. Working Group Recommendation, W3C, Feb. 2014. https://www.w3.org/TR/trig/ .

[4] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. Working Group Recommendation, W3C, Sept. 2012. http://www.w3.org/TR/r2rml/ .

[5] L. de Medeiros, F. Priyatna, and O. Corcho. MIRROR: Automatic R2RML Mapping Generation from Relational Databases. In *Engineering the Web in the Big Data Era*. 2015.

[6] T. De Nies, A. Dimou, R. Verborgh, E. Mannens, and R. Van de Walle. Enabling dataset trustworthiness by exposing the provenance of mapping quality assessment and refinement. In *Proceedings of the 4th International Workshop on Methods for Establishing Trust of (Open) Data*, 2015.

[7] A. Dimou, D. Kontokostas, M. Freudenberg, R. Verborgh, J. Lehmann, E. Mannens, S. Hellmann, and R. Van de Walle. Assessing and Refining Mappings to RDF to Improve Dataset Quality. In *Proceedings of the 14th ISWC*, 2015.

[8] 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 *Workshop on Linked Data on the Web*, 2014.

[9] A. Dimou, R. Verborgh, M. Vander Sande, E. Mannens, and R. Van de Walle. Machine-Interpretable Dataset and Service Descriptions for Heterogeneous Data Access and Retrieval. In *SEMANTiCS 2015*, 2015.

[10] O. Hartig and J. Zhao. *Provenance and Annotation of Data and Processes: Third International Provenance and Annotation Workshop, IPAW 2010*, chapter Publishing and Consuming Provenance Metadata on the Web of Linked Data. 2010.

[11] P. Heyvaert, A. Dimou, R. Verborgh, E. Mannens, and R. Van de Walle. Approaches for Generating Mappings to RDF. In *Proceedings of the 14th ISWC: Posters and Demos*, 2015.

[12] P. Heyvaert, A. Dimou, R. Verborgh, E. Mannens, and R. Van de Walle. Towards a Uniform User Interface for Editing Mapping Definitions. In *Workshop on Intelligent Exploration of Semantic Data*, 2015.

[13] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 Web Ontology Language. W3C Recom., Dec. 2012. http://www.w3.org/TR/owl2-primer/ .

[14] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M. Skjæveland, E. Thorstensen, and J. Mora. BootOX: Practical Mapping of RDBs to OWL 2. In *The Semantic Web - ISWC 2015*. 2015.

[15] M. Lanthaler. Hydra Core Vocabulary. Unofficial Draft, June 2014. http://www.hydra-cg.com/spec/latest/core/.

[16] T. Lebo, S. Sahoo, and D. McGuinness. PROV-O: The PROV Ontology. Working Group Recommendation, W3C, Apr. 2013. http://www.w3.org/TR/prov-o/ .

[17] F. Maali and J. Erickson. Data Catalog Vocabulary (DCAT). W3C Recommendation, Jan. 2014. http://www.w3.org/TR/vocab-dcat/ .

[18] L. Moreau and P. Missier. PROV-DM: The PROV Data Model. Working Group Recommendation, W3C, Apr. 2013. http://www.w3.org/TR/prov-dm/ .

[19] A.-C. Ngonga Ngomo and S. Auer. Limes: A Time-efficient Approach for Large-scale Link Discovery on the Web of Data. 2011.

[20] V. Nguyen, O. Bodenreider, and A. Sheth. Don't Like RDF Reification?: Making Statements About Statements Using Singleton Property. In *Proceedings of the 23rd International Conference on World Wide Web*, 2014.

[21] C. Pinkel, C. Binnig, E. Kharlamov, and P. Haase. IncMap: Pay As You Go Matching of Relational Schemata to OWL Ontologies. In *Proceedings of the 8th International Conference on Ontology Matching*, pages 37–48, 2013.

[22] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the Linked Data Best Practices in Different Topical Domains. In *ISWC 2014*. 2014.

[23] K. Sengupta, P. Haase, M. Schmidt, and P. Hitzler. Editing R2RML mappings made easy. 2013.

[24] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström. JSON-LD. Working Group Recommendation, W3C, Jan. 2014. https://www.w3.org/TR/json-ld/ .

[25] J. Tennison, G. Kellogg, and I. Herman. Model for Tabular Data and Metadata on the Web. W3C Working Draft, Apr. 2015. http://www.w3.org/TR/2015/WD-tabular-data-model-20150416/.

[26] R. Verborgh, O. Hartig, B. De Meester, G. Haesendonck, L. De Vocht, M. Vander Sande, R. Cyganiak, P. Colpaert, E. Mannens, and R. Van de Walle. Querying datasets on the Web with high availability. In *Proceedings of the 13th ISWC*, 2014.

[27] R. Verborgh, M. Vander Sande, P. Colpaert, S. Coppens, E. Mannens, and R. Van de Walle. Web-scale querying through Linked Data Fragments. In *Proceedings of the 7th Workshop on Linked Data on the Web*, 2014.

[28] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk – A Link Discovery Framework for the Web of Data. In *Workshop on Linked Data on the Web*, 2009.