Generating Executable Mappings from RDF Data Cube Data Structure Definitions

Christophe Debruyne[0000-0003-4734-3847], Dave Lewis[0000-0002-3503-4644], and Declan O'Sullivan[0000-0003-1090-3548]

ADAPT Centre, Trinity College Dublin, Dublin 2, Ireland
`{first.last}@adaptcentre.ie`

Abstract. Data processing is increasingly the subject of various internal and external regulations, such as GDPR which has recently come into effect. Instead of assuming that such processes avail of data sources (such as files and relational databases), we approach the problem in a more abstract manner and view these processes as taking datasets as input. These datasets are then created by pulling data from various data sources. Taking a W3C Recommendation for prescribing the structure of and for describing datasets, we investigate an extension of that vocabulary for the generation of executable R2RML mappings. This results in a top-down approach where one prescribes the dataset to be used by a data process and where to find the data, and where that prescription is subsequently used to retrieve the data for the creation of the dataset "just in time". We argue that this approach to the generation of an R2RML mapping from a dataset description is the first step towards policy-aware mappings, where the generation takes into account regulations to generate mappings that are compliant. In this paper, we describe how one can obtain an R2RML mapping from a data structure definition in a declarative manner using SPARQL CONSTRUCT queries, and demonstrate it using a running example. Some of the more technical aspects are also described.

Keywords: Data Cube, R2RML, Data Transformation

1 Introduction

The Resource Description Format [18] (RDF) provides us a flexible data model for semantic interoperability and data integration, especially in scenarios where data from various heterogeneous sources need to be processed for a particular purpose. Organizations have become increasingly sensitive to data processing policies, both organizational and especially those put forward by legislation (such as GDPR). Scholars rightfully argue that compliance should be treated in the early phases of information systems design [4]. In cases where data needs to be integrated or when a particular data processing activity was not yet foreseen, one is faced with additional challenges, such as:

- Are we allowed to process the data in a particular way?
- Are we allowed to use all the data?

- Have people given their consent for their data to be processed?

The data that is needed are often stored in different files or databases, which means they have to be retrieved and integrated. The integration of heterogenous resources can be facilitated with the Resource Description Framework (RDF), a W3C Recommendation. Mapping languages such as R2RML [6] have been used to map relational (or tabular) data to RDF datasets. R2RML can thus be used to retrieve and integrate data for the creation of a dataset for a particular data processing activity. The use of RDF allows our work to be integrated with post-hoc analysis methods reported in [16].

While the vocabularies, representations, or even formats for these datasets may be bespoke, we will adopt a standardized RDF vocabulary for representing datasets and their structure. The RDF Data Cube Vocabulary [19] is an ontology[1] for describing multi-dimensional datasets on the Semantic Web where the structure of a *dataset* can be prescribed via so-called *Data Structure Definitions* (DSDs) where *observations* are identified by their *dimensions*, capture one or more observed values via *measures*, and observed values can be annotated with *attributes*. Linked Data principles allow all these entities to be linked with other Linked Data datasets, providing one with the means to interpret or correctly process the data. Such datasets are thus curated with particular (types of) data processing in mind.

Using RDF Data Cube and R2RML[2], the steps to create a dataset would look as follows:

1. Declare a data structure definition for an application;
2. Declare a dataset which will contain the observations from a database;
3. Create a mapping from a relational database (table or query) to that dataset;
4. Execute the mapping for creating the dataset; and
5. Validate the resulting dataset.

While databases are typically used by various applications within an organizational context, particular processes often need only a subset of the data (often retrieved with a query). Similarly, the data contained in a dataset will only contain those fit for a particular purpose; e.g., the number of sales per product and week. If (parts) of the data were to be subject of a particular policy, one needs to be cautious. Since datasets are curated for a particular purpose, and those datasets may be subject to policies, it makes

[1] While RDF provides us the data model, data is usually integrated using a common model captured in a so-called ontology. Ontologies – being commonly defined as "a [formal] explicit specification of a [shared] conceptualization" [10] – are also developed for a particular purpose, but the ontologies (or vocabularies) we observe on the Linked Data Web are often lightweight and meant for information exchange. Applications that consume such Linked Data are not (necessarily) known beforehand and are often published with very accessible licenses such as Creative Commons.

[2] Even if we were not to use RDF Data Cube and R2RML, similar steps would be necessary for capturing the schema or structure of the dataset, and the creation and execution of a mapping to populate that schema.

sense to attach such policy information to the dataset or DSDs (rather than the mapping). Mappings are only concerned with retrieving and transforming the information. The problem, however, is that one tends to create a mapping manually.

The research question we aim to answer in this paper is: "Can we generate an R2RML mapping from a data structure definition?" An algorithm generating such a mapping could subsequently be extended to take into account policies so as to generate mapping that is compliant. In other words, the algorithm for generating R2RML mappings would then be "policy-aware".

The remainder of this paper is organized as follows: Section 2 presents some related work on generating datasets from relational data; Section 3 presents our approach using a running example and describes our declarative approach using SPARQL CONSTRUCT queries to generate an R2RML mapping from a DSD; Section 4 elaborates on some of the more technical aspects of our approach; Section 5 discusses some aspects of our approach; Section 6 then discusses the rationale of annotating DSDs with policy information and to extend our approach for generating customized mappings (which is part of our future work); and finally, in Section 6, we conclude our paper.

2 Related Work

To the best of our knowledge, related work in generating (R2RML) mappings from other representations is quite limited. The authors in [23] – who proposed a declarative language for ontology-based data access where a single description results in an ontology, mappings and rules for transforming queries – mentioned adopting R2RML because of its increasing uptake.

TabLinker, mentioned in [14], transforms Excel documents into Data Cube datasets by mapping the markup of cells in a Microsoft Excel's XML file to elements to a Data Cube dataset (and structure). In other words, users have to manually format the Excel file and that formatting is then used to generate the RDF dataset. OLAP2DataCube and CSV2DataCube are two tools proposed in [22] for extracting statistical data and the creation of data cube datasets. When using OLAP2DataCube, queries are mapped onto datasets, dimensions, and measures.

The Open Cube Toolkit [12] provides a D2RQ [3] extension for generating an RDF graph – according to the RDF Data Cube Vocabulary – using D2RQ's R2RML support. The D2RQ data provider requires a mapping relating a table to a dataset using a bespoke XML mapping language. The XML file – of which an example is shown in **Listing 1** – is then used to generate an R2RML mapping which is then executed by D2RQ's engine. Their approach is thus similar in that it generates an executable R2RML file from the mapping. The limitations of their approach are brought forward by their mapping language; it is bespoke, not in RDF and has not been declared in a particular namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping>
    <dataset>
        <table-name>OBSERVATIONS</table-name>
        <label>people age in Ireland</label>
        <uri>people-in-ireland</uri>
        <pattern>{"ID"}</pattern>
    </dataset>
    <dimensions>
        <dimension>
            <column>D1</column>
            <label>Age group</label>
            <uri>age-group</uri>
            <property>age-group</property>
        </dimension>
    </dimensions>
    <measures>
        <measure>
            <column>MEASURE</column>
            <label>Have a personal computer</label>
            <uri>have-a-personal-computer</uri>
            <property>have-a-computer</property>
            <datatype>xsd:int</datatype>
        </measure>
    </measures>
</mapping>
```

Listing 1. Example of the Open Cube Toolkit's mapping declaring where the various elements of a dataset can be found in a relational database table, which came out of the toolkit's box.

Wigham et al. highlighted some of the problems with the RDF Data Cube Vocabulary and proposed their own model for capturing (relational) datasets [25]. Capturing multiple observation values, for instance, is not straightforward, and they thus propose vocabulary for relational data where "cells" can be added to record tables and where records can be nested. The authors also presented a Microsoft Excel plugin for generating RDF graphs using their Record Table schema in [20]. Similar to TabLinker, the mapping from the spreadsheet to the RDF is a built-in plugin.

In [1], the author presented a tool that generated diagrams based on TURTLE. The tool is aimed to facilitate modelling by guaranteeing consistency between statements and visualization; "what you see is what you mean". In [1], the author furthermore criticizes the complexity and verbosity of R2RML, and presented an extension of the tool. By embedding SQL queries and fieldnames in the examples (in TURTLE), the tool is able to generate a complete and executable R2RML mapping.

3 R2DQB – Approach Demonstration

In this section we present R2DQB, pronounced R-2-D-cube, which is a contraction of **R2**RML and **D**ata **QB** (short for "cube").

The approach we adopt is to avail of RDF's data model to annotate data structure definitions (DSDs) (and dimensions, measures, and attributes) in such a way that DSDs are reusable, and R2RML mappings can be generated that will create a dataset "populating" the DSD. The different steps in our approach are depicted in **Fig. 1** and will be described in this section. The generated Data Cube Datasets should refer to its DSD with a `qb:structure` statement (from `qb:DataSet` to `qb:DataStructure-Defintion`), which basically informs agents that this dataset is structured according to that DSD. To exemplify our approach, we will first introduce our running example.

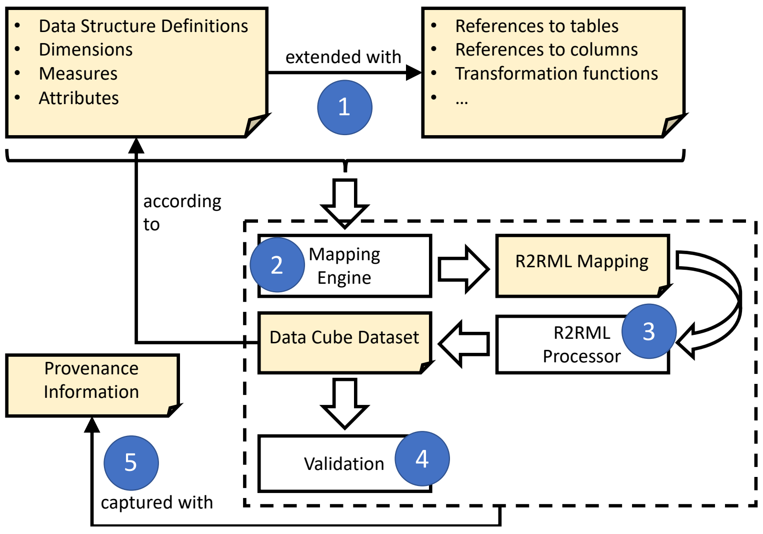

Fig. 1. The various steps in generating and executing an R2RML mapping from a DSD for the creation of a Data Cube dataset.

3.1 Running example

We have chosen to adopt the example from the RDF Data Cube Vocabulary's specification [19], shown in **Table 1**. In this table, we have examples of life expectancy (in number of years) broken down by region, age and time. There are thus three dimensions (region, time period, and gender) and one measure (life expectancy). One can also model attributes of the measure, e.g. the unit is to be interpreted as a number of years.

Table 1. Excerpt from the StatsWales report number 003311 which describes life expectancy broken down by region, age and time (from [19])

	2004-2006		**2005-2007**		**2006-2008**	
	Male	**Female**	**Male**	**Female**	**Male**	**Female**
Newport	76.7	80.7	77.1	80.9	77.0	81.5
Cardiff	78.7	83.3	78.6	83.7	78.7	83.4
Monmouthshire	76.6	81.3	76.5	81.5	76.6	81.7
Merthyr Tydfil	75.5	79.1	75.5	79.4	74.9	79.6

The example (re)uses resources (URIs) for the dimension. Albeit possible in our approach, we will first describe the steps using the values (i.e., literals) for each of the dimension. We note that according to [19], dimensions may either be a resource or a literal. We will demonstrate the use of URIs, as well as other technicalities of our approach, in a subsequent section.

3.2 Step 1: Annotating DSDs

We start off with the (re)use of a DSD including, which we will extend with information on where to fetch the data. **Listing 2** depicts an RDF graph containing a stripped-down version of the DSD from [19]. This DSD does not contain labels and links to concepts, and the range declaration in the dimensions have also been omitted as we will generate literals. **Listing 3** another RDF graph that extends the former with mapping information. We refer to a relational database table called "statssimple" with the structure depicted in **Table 2**.

Table 2. Relational database table "statssimple" for the running example.

Field	Type	Null	Key
period	varchar(20)	NO	PRI
area	varchar(20)	NO	PRI
sex	varchar(20)	NO	PRI
lifeexpectancy	float	NO	

```
@base <http://www.example.org/>
<#refPeriod> a rdf:Property, qb:DimensionProperty;
rdfs:subPropertyOf sdmx-dimension:refPeriod .

<#refArea> a rdf:Property, qb:DimensionProperty;
rdfs:subPropertyOf sdmx-dimension:refArea .

<#lifeExpectancy> a rdf:Property, qb:MeasureProperty;
rdfs:subPropertyOf sdmx-measure:obsValue;
rdfs:range xsd:decimal .

sdmx-dimension:sex a rdf:Property, qb:DimensionProperty .

<#dsd-le> a qb:DataStructureDefinition;
# The dimensions
qb:component [ qb:dimension <#refArea> ];
qb:component [ qb:dimension <#refPeriod> ];
qb:component [ qb:dimension sdmx-dimension:sex ];
# The measure(s)
qb:component [ qb:measure <#lifeExpectancy> ] .
```

Listing 2. And RDF graph describing a DSD.

```
@base <http://www.example.org/>
<#refPeriod> rr:column "period";
<#refArea> rr:column "area";
<#lifeExpectancy> rr:column "lifeexpectancy";
sdmx-dimension:sex rr:column "sex" .
<#dsd-le> rr:tableName "statssimple";
```

Listing 3. Extending the DSD of **Listing 2** with mapping information.

For this study, and to prove the feasibility of our approach, we have chosen to reuse R2RML predicates. R2RML provides us with the necessary predicates to annotate

DSDs and their components with instruction on where to find the information in a relational database. As a consequence, the graph in **Listing 3** does not constitute a valid R2RML document. A more generic approach will be considered as future work.

3.3 Step 2: The generation of an R2RML mapping

We have chosen to adopt a declarative approach to generating the R2RML mapping via a sequence of SPARQL CONSTRUCT queries. The various queries can be summarized as follows: 1) Create the triples maps (for mapping tables, views or queries to RDF); 2) Use the dimensions to create subject maps; and 3) Create predicate object maps for the measures, and dimensions.

We obtain an executable R2RML mapping by merging the models of each SPARQL CONSTRUCT query. This model is not meant to be merged with the prior RDF graphs from **Listing 2** and **Listing 3**, as it is meant to generate RDF that will be a dataset, which can be regarded as an instance of the DSD captured in **Listing 2**. In a wider governance narrative, the resulting mapping may be stored to inform stakeholders of the provenance of the datasets.

We now begin with the description of each query. We have omitted prefixes and base declarations from the queries for brevity.

The generation of a logical table for each DSD related to a table is shown in **Listing 4**. Note that views are also referred to with `rr:tableName`. A similar CONSTRUCT query is used for DSD's related to a query with the `rr:query` predicate. The namespace `pam` refers to our vocabulary developed for this study. In this listing, we create a link between triples maps and their DSDs. This will come in handy to attach the different components of the R2RML mapping (described later on). The resulting triples map is a blank node. The query can be changed to assign it an IRI, either with the IRI function or as a parameter provided by a user.

```
CONSTRUCT {
  [] rr:logicalTable [ rr:tableName ?t ] ;
     pam:correspondsWith ?x .
} WHERE {
  ?x a qb:DataStructureDefinition ;
     rr:tableName ?t .
}
```

Listing 4. Generating an R2RML triples map for each data structure definition

The CONSTRUCT query for generating the subject map is shown in **Listing 5**. Dimensions are used to identify each observation. We use that information to generate the subject map of a triples map. The columns used by the dimensions are used for creating a template that will identify each observation in the dataset. An R2RML processor will use the template, which will generate values, to keep information of each observation in an appropriate data structure (e.g., a dictionary). Next to a subject map, this mapping also creates a predicate object map that will relate individual observations to a particular dataset.

```
CONSTRUCT {
 ?tm rr:subjectMap [
  rr:class qb:Observation ;
  rr:termType rr:BlankNode ;
  rr:template ?x ;
 ] .
 ?tm rr:predicateObjectMap [
  rr:predicate qb:dataSet ;
  rr:object ?ds;
 ] .
} WHERE {
 ?tm pam:correspondsWith ?dsd ;
  rr:logicalTable [ rr:tableName ?t ] ;
 BIND(IRI(?t) AS ?ds)
 {
  SELECT
   (CONCAT("{", GROUP_CONCAT(?c; SEPARATOR="}-{"), "}") as ?x) {
   ?dsd qb:component ?component .
   { ?component qb:dimension [ rr:column ?c ] }
   UNION
   { ?component qb:dimension [
                  rrf:functionCall [
                    rrf:parameterBindings ( [ rr:column ?c ] )
                  ]
                ] }
  } GROUP BY ?dsd
 }
}
```

Listing 5. Generating an R2RML triples map for each data structure definition

Notice that we cover two cases in **Listing 5**; dimensions based on column values, and dimensions based on function calls. Function calls are used to relate column values to literals or URIs – simulating an association. We will later describe why the use of such functions is desirable.

Listing 6 provides the CONSTRUCT queries for adding object predicate maps to the triples maps based on measures. The CONSTRUCT query for a similar mapping based on dimensions is similar; one of the main differences is the predicate (highlighted in yellow). If a dimension or a measure has been declared a range, we will add that range declaration to the R2RML mapping only if that range is declared in the XSD namespace. The predicates used for dimensions and measures may both be used for literals and resources. The `rr:datatype` predicate of R2RML is only used for data types (literals). If we were to remove the filter, we could end up with predicate object maps that generate resources *and* are provided a datatype. The R2RML specification states that this is erroneous.

A current limitation is that we can only deal with XSD datatypes. Mappings using datatypes outside of XSD, such as `geo:wktLiteral` for polygons for geospatial data in GeoSPARQL [15], cannot yet be generated. A naïve approach would be to keep track of an exhaustive list of datatypes in the query, but more elegant approaches will be investigated as part of our future work.

```
CONSTRUCT {
 ?tm rr:predicateObjectMap [
  rr:predicate ?measure ;
  rr:objectMap [
   ?p ?c ; rr:termType ?type ; rr:datatype ?range ;
  ] ;
 ]
} WHERE {
 ?tm pam:correspondsWith ?dsd .
 ?dsd qb:component ?component .
 ?component qb:measure ?measure .
 ?measure ?p ?c .
 ?measure rr:column|rr:template|rr:constant|rrf:functionCall ?c .
 OPTIONAL { ?measure rr:termType ?type }
 OPTIONAL {
  ?measure rdfs:range ?range .
  FILTER(
    CONTAINS(STR(?range), "http://www.w3.org/2001/XMLSchema#")
  )
 }
}
```

Listing 6. Generating predicate object maps from measures

Finally, the generated dataset also needs to be connected to its DSD. This is straightforward with the following CONSTRUCT query (in **Listing 7**). The IRI of the dataset is based on the table's name (or query). Some relational databases allow spaces (or other special characters) to be used in table names. The function `ENCODE_FOR_URI` ensures that characters in table names (such as spaces in some databases) are correctly encoded.[3]

```
CONSTRUCT {
  ?ds a qb:DataSet ; qb:structure ?x .
 } WHERE {
  ?x a qb:DataStructureDefinition ;
     rr:tableName ?t .
  BIND(IRI(ENCODE_FOR_URI(?t)) AS ?ds)
 }
```

Listing 7. Creating an instance of a dataset based on an annotated DSD

With these mappings – which are declarative and implemented as SPARQL CONSTRUCT queries – we are able to generate an executable R2RML mapping. Given our table "statssimple" (see **Table 2**) and the snippets from **Listing 2**, the R2RML in **Listing 8** is generated. While it is not explicit that the resource is a `rr:TriplesMap`, it will be inferred by the R2RML engine as such due to the domain of `rr:logicalTable` being `rr:TriplesMap`.

```
[ pam:correspondsWith    <http://www.example.org/#dsd-le> ;
  rr:logicalTable        [ rr:tableName  "statssimple" ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [ rr:column  "area" ] ;
    rr:predicate  <http://www.example.org/#refArea>
```

[3] We thank the anonymous reviewer for spotting this issue.

```
  ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [ rr:column  "period" ] ;
    rr:predicate  <http://www.example.org/#refPeriod>
  ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [ rr:column  "sex" ] ;
    rr:predicate  sdmx-dimension:sex
  ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [
      rr:column    "lifeexpectancy" ;
      rr:datatype  xsd:decimal
    ] ;
    rr:predicate  <http://www.example.org/#lifeExpectancy>
  ] ;
  rr:predicateObjectMap  [
    rr:object     <statssimple> ;
    rr:predicate  qb:dataSet
  ] ;
  rr:subjectMap  [
    rr:class     qb:Observation ;
    rr:template  "{area}-{period}-{sex}" ;
    rr:termType  rr:BlankNode
  ]
] .
```

Listing 8. Generated R2RML

3.4 Step 3: Execution of the R2RML mapping

For the execution of our mapping, we rely on an implementation of the R2RML implementation developed by the ADAPT Centre.[4] This implementation complies with the R2RML implementation as it is used to demonstrate minimal extensions of the mapping language (such as functions written in JavaScript [7]). The mapping in **Listing 8** contains no statements that fall outside R2RML's scope and should work with other implementation of the specification. This mapping execution generated 144 triples; 6 triples for each of the 24 observations. An example of such an observation is shown in **Listing 9**.

```
[ a                                       qb:Observation ;
qb:dataSet                              <statssimple> ;
sdmx-dimension:sex                      "Male" ;
<http://www.example.org/#lifeExpectancy>  78.6 ;
<http://www.example.org/#refArea>         "Cardiff" ;
<http://www.example.org/#refPeriod>       "2005-2007"      ] .
```

Listing 9. An observation generated with the R2RML mapping of **Listing 8**

In the case of an XSD datatype, our R2RML processor checks whether a value that is generated by an object map corresponds with that datatype and reports when this is not the case. When a datatype is not part of the XSD namespace is used for an object

[4] https://github.com/chrdebru/r2rml

map, such as `ex:myInteger` for instance, the literal is merely typed with that datatype. If no datatype is provided, the datatype of the literal depends on the datatype of the column (see Section 10.2 "10.2 Natural Mapping of SQL Values" of [6]).

3.5 Step 4: Validating the generated RDF

We validate the generated RDF by checking the integrity constraints put forward by the RDF Data Cube Vocabulary, presented as a set of so-called integrity constraints in the specification [19]. This is necessary as the execution of the R2RML mapping does not guarantee a well-formed cube. R2RML prescribes, for instance, that a triple is not added to the model if any of the columns used for a predicate, object or subject contains a NULL value. If a cell for a dimension were to have a NULL value, then no triple is generated, but an observation must be related to all dimensions.

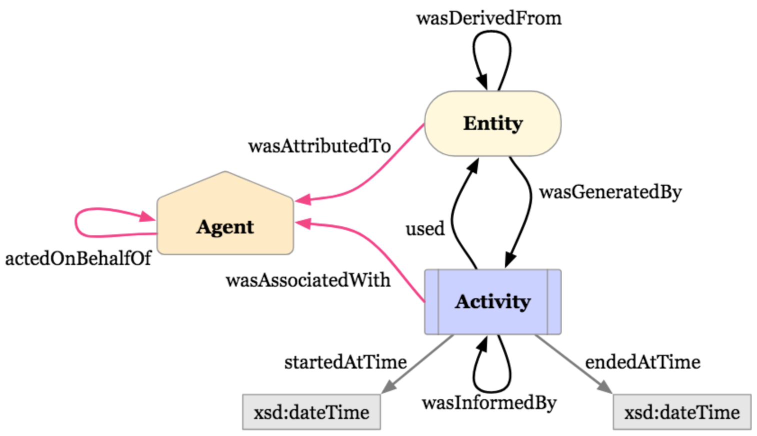

Fig. 2. Core concepts and relations in PROV-O from [21], Copyright © 2011-2013 W3C® (MIT, ERCIM, Keio, Beihang).

3.6 Step 5: Provenance information

We note that Step 5 is not really a step per se, but that the process captures provenance information during all aforementioned steps. Provenance information provides insights on a resource's origin, such as who created that resource, when it was modified or how it was created [26]. PROV-O [21], which we have adopted for this study, is a W3C Recommendation for representing and exchanging provenance information as RDF. PROV-O's core concepts and relations (shown in **Fig. 2**) provide a good starting point for describing the activities and intermediate artifacts that lead to the realization of an ontology mapping.

Rather than providing a snippet of the generated RDF, we will describe how we extended PROV-O and how the entities are used an interrelated. The classes we have declared in our namespace and which PROV-O concepts they specialize are shown in **Table 3**. Our proof-of-concept relies on R2RML-F, which will be an instance of pam:R2RML_Processor. Our pam:Mapping_Generator is our implementation of D2RQB. We also developed an instance of pam:Validator, which currently implements the integrity constraints prescribed by [19].

Table 3. Classes that extend PROV-O

Classes extending prov:Entity	
pam:DSD_Document	Used to represent RDF documents/graphs containing our annotated Data Structure Definitions.
pam:R2RML_Mapping	Used to represent the generated R2RML mappings
pam:Dataset	Used to represent the generated Data Cube datasets
pam:Validation_Report	Used to represent the validation reports
Classes extending prov:Activity	
pam:Generate_Mapping	Represents the activity of generating an R2RML mapping from an annotated DSD
pam:Execute_Mapping	Represents the activity of executing the R2RML mapping
pam:Validate_Dataset	Represents the activity of validating a mapping
Classes extending prov:(Software)Agent	
pam:Mapping_Generator	Represents the software agent that generates an R2RML mapping as per approach.
pam:R2RML_Processor	Represents the software agent executing the mapping
pam:Validator	Represents the software agent validating the dataset

Fig. 2 clearly depicts how the main entities of PROV-O are interrelated. The relations between our entities are as follows:

- A pam:Generate_Mapping uses (prov:uses) a pam:DSD_Document to generate a pam:R2RML_Mapping. A mapping is thus generated by (prov:wasGeneratedBy) such an activity. This activity was performed (prov:wasAssociatedWith) by our implementation of our approach (pam:Mapping_Generator). The mapping is also derived from the pam:DSD_Document, so we also assert a prov:wasDerivedFrom between the two entities.
- A pam:Execute_Mapping uses (prov:uses) a pam:R2RML_Mapping to generate a pam:Dataset. That dataset was thus generated by (prov:wasGeneratedBy) that activity. This activity was performed by (prov:wasAssociatedWith) by the pam:R2RML_Processor we adopted.
- A pam:Validator uses (prov:uses) both a generated dataset (pam:Dataset) and DSD document (pam:DSD_Document) for validating the dataset and producing a report (pam:Validation_Report). It relies on an implementation (prov:wasAssociatedWith a pam:Validator) of at least the integrity constraints prescribed by [19].

We also store timestamps (start and end-time) of each activity. The adoption of PROV-O in this study allows us to create traceable data flows where a DSD can be used to generate an executable R2RML document multiple times. This helps us fulfill some of the requirements put forward by policies.

4 Extended Demonstration

In the previous section we demonstrated our approach using a running example. In this section, we demonstrate more advanced aspects of our approach starting from the same example.

4.1 Mapping values onto URIs

A common mapping problem is relating column values to a corresponding set of "values" (IRIs or literals); for instance "blauw" corresponds with `dbpedia:Blue`, "rood" with `dbpedia:Red`. R2RML provides no support for capturing such correspondences as part of the mapping. One can create a mapping from an SQL query in which an SQL CASE function is used to relate column values to, for instance, IRIs, but these then become quite cumbersome to maintain. The D2R mapping language [2] provided a convenient way to relate these correspondences via a so-called "translation table". In our approach, we adopted a minimal extension of R2RML called R2RML-F with support [7] for functions. R2RML-F allows us to create a function that takes as input a column-value and returns the corresponding value. We then still are able to benefit from keeping the table or query to be mapped as simple as possible and – if need be – only change the function to deal with changes in the ontology or source data. We provide more detail on how these functions look like in the next subsection.

4.2 Inclusion of data transformation functions

The example of [19] proposes to use the data.gov.uk reference time service to represent the time period. The following URI represents a 3-year period starting from the 1st of January, 2004: `http://reference.data.gov.uk/id/gregorian-interval/2004-01-01T00:00:00/P3Y` We can use this knowledge to extract the start year from our period and fill in a template to generate such an IRI in our DSD, as shown in **Listing 10**. One can easily see how correspondences described in Section 4.1 can then be implemented using conditions.

```
<#refPeriod> a rdf:Property, qb:DimensionProperty;
  rrf:functionCall [
    rrf:function _:b0 ;
    rrf:parameterBindings ( [ rr:column "period" ] ) ;
  ]                                                                   ;
  rr:termType rr:IRI ;
  rdfs:label "reference period"@en;
  rdfs:subPropertyOf sdmx-dimension:refPeriod;
  rdfs:range interval:Interval;
  qb:concept sdmx-concept:refPeriod
  .
_:b0
  rrf:functionName "translateperiod" ;
  rrf:functionBody """
    function translateperiod(var1) {
      return "http://reference.data.gov.uk/id/gregorian-interval/" +
             var1.substring(0, 4) + "-01-01T00:00:00/P3Y"
      }
    """                                                                        ;
.
```

Listing 10. Using functions in a DSD to transform data

The example in **Listing 10** only relies on string manipulation. While seemingly simple, the example serves to demonstrate a particular aspect of our approach. [17] argued why and when support for functions in mappings are desirable; e.g., when the underlying (database) technology provides no support for data transformation.

4.3 Interlinking datasets

Both dimensions and measures may refer to either literals or resources. The use of resources (typically identified by a URI) are common for dimensions, but less so for measures. One measures values that you want to manipulate, compare, etc. In our approach, this is feasible by creating predicate object maps that generate resources (with a URI). By changing

```
<#refArea> rr:column "area";
sdmx-dimension:sex rr:column "sex";
```

in **Listing 2** into

```
<#refArea> rr:template "http://example.org/area/{area}";
sdmx-dimension:sex rr:template "http://dbpedia.org/resource/{sex}";
```

we create resources for areas and gender. The first generates "local" resources as `http://example.org/` is the namespace of our running example. The second generates DBpedia [13] URIs. If the second is used, not only does one create links across datasets, but also with other Linked Data datasets as well.

5 Discussion

While our approach adopted R2RML for the mapping, adoption of other R2RML dialects such as RML [8] and xR2RML [11] is feasible. These implementations provide native support for other formats such as CSV and JSON. The R2RML processor we have adopted allows us to approach non-relational data as such by either treating tabular data files as relational tables (without keys) or by formulating SQL queries for NoSQL databases by means of Apache Drill[5]. As long as the DSD is annotated with the names of tables, views and fields that appear in the database, an executable R2RML mapping can be produced. Verifying whether those annotations are correct (e.g., does the table exist) falls outside the scope of this paper. Similarly, verifying whether the resulting dataset complies with the DSD or the ontologies used by the DSD are up to the Data Cube validator and external tools respectively. The R2RML processor may be in charge of verifying whether values comply with datatypes, but the Recommendation does not require this functionality.

[5] `https://drill.apache.org/`

W3C has published Recommendations for representing the “schemas” of tabular data the Web [17] and how to generate RDF from those [24]. The goals of these initiatives were to standardize access to information in tabular form, to propose a schema language for tabular data, and to specify the conversion of such data into RDF (amongst others) as part of that initiative. Their goal was thus not to represent datasets in RDF. The advantage of [17] is that its schema language does provide straightforward value constraints and the investigation of this vocabulary and converters as an alternative might be worthwhile investigating.

This study focuses on the generation of an R2RML file for creating RDF Data Cube Datasets. We note that the RDF Data Cube Vocabulary allows one to publish multi-dimensional data *in general*. While the vocabulary’s underlying model is indeed an ISO standard for representing statistical data and its metadata, the vocabulary is generic enough to represent even simple “relational” data and datasets fit for training Machine Learning models. As it is capable of representing a wide variety of datasets and is – unlike other vocabularies – standardized, we deemed it the most suitable for our study.

6 Towards a policy-aware mapping engine

Now that we have demonstrated the feasibility of annotating a data structure definition such that an executable R2RML mapping can be generated, we can elaborate on our vision towards a policy-aware mapping engine.

We stated that datasets are created for a particular purpose. Those purposes are not necessarily known beforehand and have to be created. Regulations – both internal as well as external (such as GDPR) – may require that the data contained in datasets and data processing complies with these regulations. In the case of GDPR, for instance, users have to be informed how their data is used, and they also have to give their consent.

The next step in our research is thus to tackle the problem of generating datasets which ensure that compliance. Given a knowledge base containing formalized descriptions of regulations, informed consent, etc., how can we adapt our mapping engine (number 2 in **Fig. 1**) as to generate executable mappings that generate compliant datasets? In [9], the authors propose a semantic model for expressing consent leveraging existing semantic models of provenance, processes, permission and obligations. We may be able to use this as a basis for formalizing data processing purposes and the data used in that processing. Knowing the importance of regulatory compliance, such an approach renders compliance checking a flexible, adaptable top-down approach to data processing.

7 Conclusions and Future Work

The increasing pressure for organizations to be compliant with various regulations and policies provided the motivation for this study. As organizations need to demonstrate that their data processing activities (which evolve over time) are compliant (e.g., GDPR), they can benefit from semi-automated processes that facilitate compliance processes. In this study, we argued that data processing activities rely on datasets. A data process will rarely need all the data stored in one or more data sources. W3C put forward a Recommendation for describing the structure and capturing multi-dimensional datasets, called the RDF Data Cube vocabulary. This allows one to declare how a data set should look like for a particular data processing activity. In this study, we aimed to answer the following research question: "How can we generate an R2RML mapping from a data structure definition?" in order to create the datasets.

We have proposed a declarative approach to generating an R2RML mapping from a Data Set Structure definition by 1) annotating the DSD with some predicates based on R2RML, and 2) executing a sequence of SPARQL CONSTRUCT query that generates the R2RML mapping. The demonstration in our study shows that our approach is viable, and even highlighted some of the limitation of R2RML (data transformation functions and syntactic sugar for correspondences). Our approach is furthermore built on top of PROV-O, a provenance ontology, as to ensure the traceability of data processing activities. While this study limits itself to the generation of provenance information of our prototype, PROV-O can be immediately used to relate our generated datasets with data processing activities. This is for instance demonstrated in [16], where PROV-O is used to validate the informed consent for data processing activities.

The next step in our research is thus to tackle the problem of generating datasets which ensure that compliance. While studies like [17] investigate the use of PROV-O to check compliance "post-hoc" basis or based on a questionnaire, we will investigate a "policy-aware" mapping generator.

Acknowledgements

The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund

References

1. Alexiev, V. RDF By Example: rdfpuml for true RDF diagrams, rdf2rml for R2RML generation. In: Semantic Web in Libraries 2016 (2016), http://vladimiralexiev.github.io/pres/20161128-rdfpuml-rdf2rml/
2. Bizer, C.: D2R MAP - A database to RDF mapping language. In: King, I., Maˊray, T. (eds.) Proceedings of the Twelfth International World Wide Web Conference - Posters, WWW 2003, Budapest, Hungary, May 20-24, 2003 (2003)
3. Bizer, C., Seaborne, A.: D2RQ - treating non-RDF databases as virtual RDF graphs. In: ISWC2004 (posters) (November 2004), http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/Bizer-D2RQ-ISWC2004-Poster.pdf
4. Bonazzi, R., Hussami, L., Pigneur, Y.: Compliance Management is Becoming a Major Issue in IS Design, pp. 391–398. Physica-Verlag HD, Heidelberg (2010). https://doi.org/10.1007/978-3-7908-2148-2 45

5. Crotti Junior, A., Debruyne, C., Brennan, R., O'Sullivan D.: An evaluation of uplift mapping languages. IJWIS 13(4): 405-424 (2017)
6. Das, S., Cyganiak, R., Sundara, S.: R2RML: RDB to RDF mapping language. W3C Recommendation, W3C (Sep 2012), http://www.w3.org/TR/2012/REC-r2rml-20120927/
7. Debruyne, C., O'Sullivan, D.: R2RML-F: towards sharing and executing domain logic in R2RML mappings. In: Auer, S., Berners-Lee, T., Bizer, C., Heath, T. (eds.) Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co- located with 25th International World Wide Web Conference (WWW 2016). CEUR Workshop Proceedings, vol. 1593. CEUR-WS.org (2016), http://ceur-ws.org/Vol- 1593/article-13.pdf
8. Dimou, A. Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., and Van de Walle, R.: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Bizer, C., Auer, S., Berners-Lee, T., Heath, T. (eds.) Proceedings of the Workshop on Linked Data on the Web, LDOW 2014 - co-located with the 23rd International World Wide Web Conference (WWW 2014). CEUR Workshop Proceedings, vol. 1184. CEUR-WS.org (2014) http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf
9. Fatema, K., Hadziselimovic, E., Pandit, H.J., Debruyne, C., Lewis, D., O'Sullivan, D.: Compliance through informed consent: Semantic based consent permission and data management model. In: Brewster, C., Cheatham, M., d'Aquin, M., Decker, S., Kirrane, S. (eds.) Proceedings of the 5th Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn2017) co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 22, 2017. CEUR Workshop Proceedings, vol. 1951. CEUR-WS.org (2017), http://ceur- ws.org/Vol-1951/PrivOn2017_paper_5.pdf
10. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? International Journal on Human-Computer Studies. 43(5-6), 907–928 (1995). https://doi.org/10.1006/ijhc.1995.1081
11. Michel, F., Djimenou, L., Faron-Zucker, C., and Montagnat, J.: Translation of Relational and Non-relational Databases into RDF with xR2RML. In: Monfort, V., Krempels, K.-H., Majchrzak, T. A., and Turk Z. (eds.) WEBIST 2 015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies. SciTePress, 443-454 (2015)
12. Kalampokis, E., Nikolov, A., Haase, P., Cyganiak, R., Stasiewicz, A., Karamanou, A., Zotou, M., Zeginis, D., Tambouris, E., Tarabanis, K.A.: Exploiting linked data cubes with opencube toolkit. In: Horridge, M., Rospocher, M., van Ossenbruggen, J. (eds.) Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014. CEUR Workshop Proceedings, vol. 1272, pp. 137–140. CEUR- WS.org (2014), http://ceur-ws.org/Vol-1272/paper 109.pdf
13. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web 6(2), 167– 195 (2015). https://doi.org/10.3233/SW-140134
14. Meroo-Peuela, A., Hoekstra, R., Guret, C., Schlobach, S.: Detecting and reporting extensional concept drift in statistical linked data. In: Capadisli, S., Cotton, F., Cyganiak, R., Haller, A., Hamilton, A., Troncy, R. (eds.) Proceedings of the 1st International Workshop on Semantic Statistics (SemStats). No. 1549 in CEUR Workshop Proceedings, Aachen (2013), http://ceur-ws.org/Vol-1549/#article-10
15. Open Geospatial Consortium. GeoSPARQL - A Geographic Query Language for RDF Data. OGC (2012) http://www.opengeospatial.org/standards/geosparql
16. Pandit, H.J., Lewis, D.: Modelling provenance for GDPR compliance using linked open data vocabularies. In: Brewster, C., Cheatham, M., d'Aquin, M., Decker, S., Kirrane, S. (eds.)

Proceedings of the 5th Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn2017) co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 22, 2017. CEUR Workshop Proceedings, vol. 1951. CEUR-WS.org (2017), http://ceur-ws.org/Vol- 1951/PrivOn2017 paper 6.pdf

17. Pollock, R., Tennison, J., Kellogg, G., and Herman, I.: Metadata Vocabulary for Tabular Data. W3C Recommendation, W3C (Dec 2015) https://www.w3.org/TR/2015/REC-tabular-metadata-20151217/
18. Raimond, Y., Schreiber, G.: RDF 1.1 Primer. W3C note, W3C (Jun 2014), http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
19. Reynolds, D., Cyganiak, R.: The RDF data cube vocabulary. W3C Recommendation, W3C (Jan 2014), http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/
20. Rijgersberg, H., Wigham, M., Top, J.L.: How semantics can improve engineering processes: A case of units of measure and quantities. Advanced Engineering Informatics 25(2), 276–287 (2011). https://doi.org/10.1016/j.aei.2010.07.008
21. Sahoo, S., McGuinness, D., Lebo, T.: PROV-o: The PROV ontology. W3C recommendation, W3C (Apr 2013), http://www.w3.org/TR/2013/REC-prov-o- 20130430/
22. Salas, P.E.R., Mota, F.M.D., Breitman, K.K., Casanova, M.A., Martin, M., Auer, S.: Publishing statistical data on the web. Int. J. Semantic Computing 6(4), 373–388 (2012). https://doi.org/10.1142/S1793351X12400119
23. Skjaeveland, M. G., Giese, M., Hovland, D., Lian, E. H., and Waaler, A.: Engineering ontology-based access to real-world data sources. J. Web. Sem. 33, 112-140 (2015) https://doi.org/10.1016/j.websem.2015.03.002
24. Tandy, J., Herman, I., and Kellogg, G.: Generating RDF from Tabular Data on the Web. W3C Recommendation, W3C (Dec 2015) https://www.w3.org/TR/2015/REC-csv2rdf-20151217/
25. Wigham, M., Rijgersberg, H., de Vos, M., Top, J.: Semantic support for tables using rdf record table. International Journal on Advances in Intelligent Systems 8(1-2), 128–144 (2015)
26. Zhao, J., Hartig, O.: Towards interoperable provenance publication on the linked data web. In: Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M. (eds.) WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012. CEUR Workshop Proceedings, vol. 937. CEUR-WS.org (2012), http://ceur- ws.org/Vol-937/ldow2012-paper-03.pdf