"Just-in-Time" Generation of Datasets by Considering Structured Representations of Given Consent for GDPR Compliance

Christophe Debruyne[0000-0003-4734-3847], Harshvardhan J. Pandit[0000-0002-5068-3714], Dave Lewis[0000-0002-3503-4644], and Declan O'Sullivan[0000-0003-1090-3548]

ADAPT Centre, Trinity College Dublin, Dublin 2, Ireland
`{debruync,pandith,dave.lewis,declan.osullivan}@tcd.ie`

Abstract. Data processing is increasingly becoming the subject of various policies and regulations, such as the General Data Protection Regulation (GDPR) that came into effect in May 2018. One important aspect of GDPR is informed consent, which captures one's permission for using one's personal information for specific data processing purposes. Organizations must demonstrate that they comply with these policies. The fines that come with non-compliance are of such importance that it has driven research in facilitating compliance verification. The state of the art primarily focuses on, for instance, the analysis of prescriptive models and posthoc analysis on logs to check whether data processing is compliant to GDPR. We argue that GDPR-compliance can be facilitated by ensuring datasets used in processing activities are compliant with consent from the very start. The problem addressed in this paper is how we can generate datasets that comply with given consent "just-in-time". We propose RDF and OWL ontologies to represent the consent an organization has collected and its relationship with data processing purposes. We use this ontology to annotate schemas, allowing us to generate declarative mappings that transform (relational) data into RDF driven by the annotations. We furthermore demonstrate how we can create compliant datasets by altering the results of the mapping. The use of RDF and OWL allows us to implement the entire process in a declarative manner using SPARQL. We have integrated all components in a service that furthermore captures provenance information for each step, further contributing to the transparency that is needed towards facilitating compliance verification. We demonstrate the approach with a synthetic dataset simulating users (re-)giving, withdrawing, and rejecting their consent on data processing purposes of systems. In summary it is argued that the approach facilitates transparency and compliance verification from the start, reducing the need for posthoc compliance analysis common in the state of the art.

Keywords: GDPR - Consent - Data Integration - Semantic Mappings

1 Introduction

As noted in [1]: *"computing professionals involved in "big data" research should pay attention if they wish to gain access to datasets containing or derived from personal information."* While the article was written for data science researchers, this statement is true for all data processing purposes. Data processing, in general, is increasingly the subject of regulations, such as the General Data Protection Regulation[1] (GDPR). Such initiatives spur the investment in means and resources that will in novel ways facilitate compliance verification [2]. These initiatives are motivated by the fines that come with being non-compliant, such as up to 4% of an organization's global revenue for GDPR.

Semantic and Linked Data technologies have proven that they can facilitate interoperability, transparency, and traceability in organizations [3]. This is why these technologies are being adopted to facilitate GDPR compliance verification processes. Our prior [2, 4–6] and related [7, 8] work addressed problems of representing aspects of GDPR [2, 4, 6], the various interactions between stakeholders [2, 6] and representing (informed) consent [2, 5, 9]. Only a few studies, including ours, demonstrate how this information can be rendered actionable, namely: [5] maps consent information to eXtensible Access Control Markup Language (XACML) for managing access to data; and SPECIAL [8] analyses the compliance of a process before execution. Rather than analyzing the processes (before or after execution), we want to investigate whether an organization can also use this information

[1] https://eur-lex.europa.eu/eli/reg/2016/679/oj

for the creation of a compliant dataset, therefore guaranteeing a compliant execution of a process. The problem we address in this article is thus how to generate datasets "just-in-time" (JIT) that are compliant with certain regulations such as the consent information required by GDPR.

We motivate the use of the Resource Description Framework (RDF) for the following reasons. First, organizations often hold data in various heterogeneous data sources, motivating the need for a common graph-based data model, and for standardized technologies for transforming non-RDF data in RDF (called "uplift"). Note also that transforming RDF into non-RDF (called "downlift") is often straightforward (e.g., [10] for RDF to CSV, and [11] for RDF to XML). Secondly, with RDF and its surrounding technologies (query languages, reasoning capabilities, provenance models, etc.) being already standardized and well established, one can easily avail of existing tooling. Python, R, and Java are all popular programming languages with mature libraries for processing RDF –with Python and R being popular for data analysis.

We generalize, extend, and integrate ideas previously reported in [12] and [9]. Both studies looked into an RDF representation of multi-dimensional statistical datasets using the RDF Data Cube Vocabulary [13]. [12] presented a method for generating R2RML mappings that generate such datasets using Data Structure Definitions (DSDs). In [9], we demonstrated the feasibility of generating RDF Data Cube datasets that consider the consent an organization has gathered. In this article, we:

- integrate both approaches;
- introduce support for representing tabular data [11], thereby demonstrating its genericity; and
- adopt and extend the Data Privacy Vocabulary (DPV) [14] vocabulary rather than our bespoke ontologies to integrate the various data sources.

The remainder of this article is organized as follows. We formulate the objectives of this study in Section 2. In Section 3, we present the design of our solution and briefly elaborate on the various components: representing consent and consent information, generating mappings from annotated schemas, and the generation of datasets for GDPR compliance. Those three components are elaborated in more detail in Sections 4, 5, and 6 respectively. To demonstrate our approach, we avail of a running example throughout these three sections. We describe, in Section 7, the motivation for supporting tabular data next to RDF Data Cubes, and elaborate on the differences between the approach for tabular data introduced in this paper and the approach for RDF Data Cubes introduced in [9]. Section 8 presents the service that we developed. We conclude the paper in Section 9.

2 Objectives, Scope, and Assumptions

The objective of this study is to propose a method for generating datasets in a "GDPR-aware" manner, with a focus on consent and fit for particular data processing purposes. We hypothesize that such an approach would facilitate compliance verification, facilitate transparency, and reduce the need for posthoc compliance analysis. We break down this objective into the following specific objectives:

1. To design and develop a model for representing the consent gathered by an organization and its relations to a system's data processing purposes.
2. To propose a method for annotating schemas for the generation of compliant datasets "just-in-time".
3. To develop the various processes in our approach in a declarative manner, adopting standardized vocabularies and techniques to maximize transparency.
4. To integrate the various components in a proof-of-concept service for the demonstration and evaluation of our approach.

We assume the consent (and the information thereof) an organization has stored is valid. In other words, we assume that the consent collected and stored satisfies all the requirements for valid consent under GDPR[2]. How that consent was obtained is outside the scope of this paper. We also note that we are primarily concerned with the generation of datasets for data processing purposes. Compliance verification of activities when data is to be shared with third parties is not within the scope of this article.

3 Design

We decompose the design and development of our solution in different components: an ontology for representing consent, the generation of *uplift*[3] mappings, and the component that uses these two for the generation of compliant datasets. In GDPR, the term "purpose" refers to "the aim or goal towards which the data is processed (or associated with any other form of action)" [6]. **Fig. 1** depicts the various processes (rectangles) and artifacts (dog-eared rectangles) in our architecture. The figure furthermore depicts an organization's databases that are involved as well as a knowledge base containing consent information.

- The "**consent ontology**", shown in **Fig. 1** as an artefact with a red outline, is presented in Section 4**.** We stated in the introduction that prior work relied on bespoke ontologies that we had developed. In this article, we have adopted the Data Privacy Vocabulary (DPV) [14], which is an initiative by the W3C Data Privacy Vocabularies and Controls Community Group to specify an ontology of terms for annotating personal data handling in the context of GDPR. As the community aims to standardize this vocabulary, we expect our proposed solution will have greater uptake by adopting and, extending this ontology as needed. Our approach extends the ontology by integrating the provenance ontology PROV-O [15] for the evolution of consent and with additional predicates for annotating and interlinking data schemas with DPV (see later). Our consent ontology is used to annotate a dataset schema description with a data processing purpose as well as to populate a knowledge base with previously gathered consent. That knowledge base will be key in the "just-in-time" creation of a GDPR-compliant dataset (shown in pink).

- In green, we show the processes and artefacts involved in the **generation of uplift mappings** and the execution of those mappings to generate datasets. To generate those by the mapping engine, we adopt vocabularies to represent datasets and annotate those with references to tables, columns, and so on. The mapping engine will process both the schema and the annotations to generate an R2RML [16] mapping, a W3C Recommendation for declaring how relational data should be transformed into RDF. The R2RML mapping is then executed by an R2RML processor, producing a dataset. The datasets will, itself, be validated against the dataset's schema. We elaborate on this process in Section 5.

- In pink, we show the processes, artefacts, and consent knowledge base involved in the **creation of a compliant dataset** process. This component interrogates the previously generated dataset and the consent knowledge base to filter out all information that should not be included in the dataset. We motivate filtering the dataset after the execution of the mapping in Section 6.

We will cover related work from the state of the art related to each component in each respective subsection. The service that integrates all the various processes is presented in Section 8.

[2] https://www.privacy-regulation.eu/en/article-4-definitions-GDPR.htm, last accessed January 2020

[3] Uplift is the process of generating RDF from non-RDF resources [31]

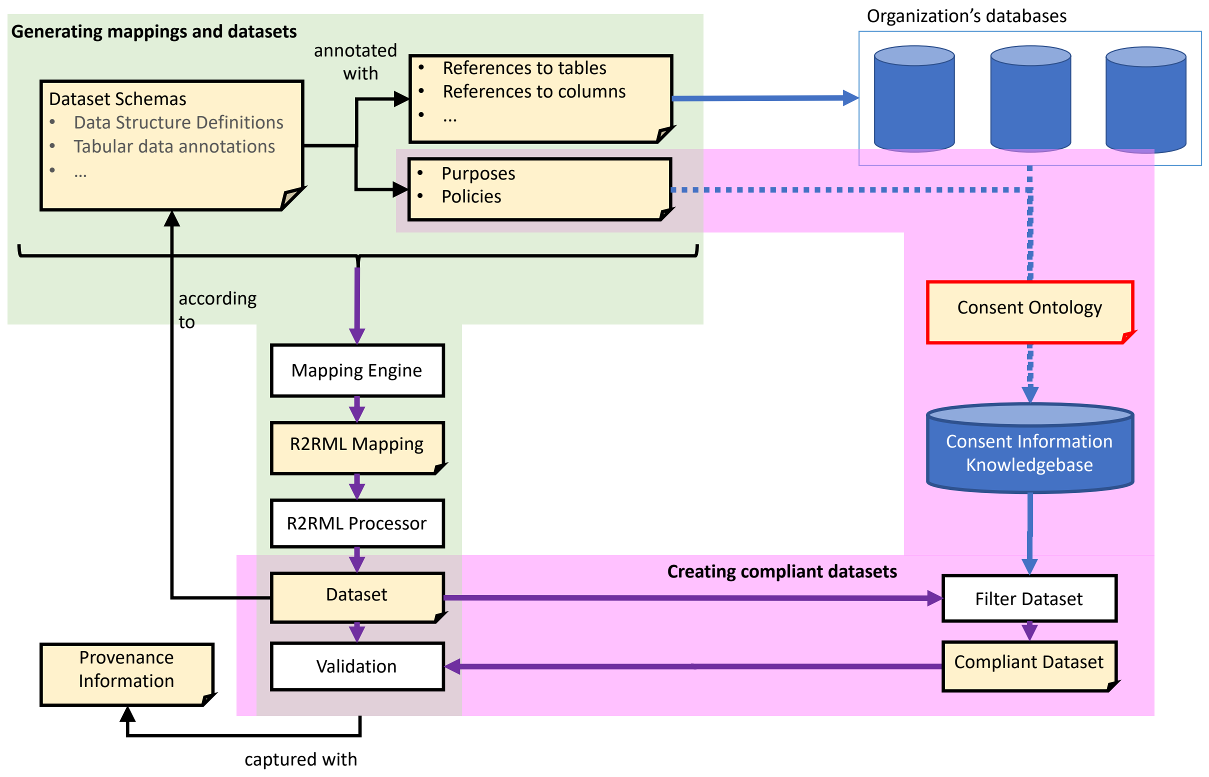


Fig. 1. Overview of artifacts and processes in our architecture. Dog-eared rectangles represent artifacts and rectangles represent processes. The consent ontology (an artifacts) is used to populate a consent information knowledge base from an organization's databases (not necessarily relational).

We stated in the introduction that prior work, presented in [12] and [9], looked into the generation of R2RML from annotated Data Structure Definitions (DSDs) and the generation of compliant datasets from these R2RML mappings. A DSD can be regarded as a schema for (statistical) datasets, and a DSD can be described with the RDF Data Cube Vocabulary [13] specification. The use of this vocabulary for representing datasets may, at times, become fairly complex for "simple" tabular or relational data. For this reason, we have extended our approach to supporting tabular data represented as RDF using a model for tabular data on the Web – CSVW [11]. In Section 7, we will discuss the challenges of using the RDF Data Cube vocabulary in our approach. The adoption of tabular data not only simplified the data that one can use for data processing, but the tabular model also reduced the number of steps for generating an R2RML mapping. Section 8 will thus also present the differences in the generation of R2RML and compliant datasets. One of the key differences between RDF Data Cube and CSVW is expressiveness. In RDF Data Cube, one needs to explicitly indicate which values identify "records" and it includes the predicates for doing so. In CSVW, it is not mandatory to declare a schema in which you indicate the columns identifying a record. We thus need to provide a way to declare such columns in our approach. We will prescribe the use of `dct:identifier` as a special predicate for indicating these columns, which will be explained later on.

4 Representing Consent Information

In November 2019, the W3C Data Privacy Vocabularies and Controls Community Group published the first version of their Data Privacy Vocabulary [14] or DPV. DPV is used to describe personal data handling. The ontology we developed for prior work (in [9]) was built from the perspective of an organization and only concerned with whether consent was given by data subjects . While our prior work [9] had most of the concepts and relations we need for our study, our proposed solution will likely have more impact by adopting the appropriate concepts and relations from DPV instead, given that the W3C Community group aims to standardize the vocabulary. The concepts and relations we needed to include in our consent ontology are mostly related to the appropriate DPV ones and also linking dataset schemas with an organization's data.

The main concepts and interrelations of the model are shown in **Fig. 2** (modeled with Graffoo [17]). We note that both the domain of `dpv:hasPurpose` and the range of `dpv:hasDataSubject` is the union of `dpv:Consent` and `dpv:PersonalDataHandling`. For this article, however, we are only concerned with `dpv:Consent` and have omitted the concept-disjunction[4] *from the Figure*, for the reader. To further simplify the Figure for the reader, we also depict the range of provision-, withdrawal-, and expiry times to be `time:Instant` instead of its superclass `time:TemporalEntity`. While the time ontology also provides constructs to model durations, we only use date and time instances in this study.

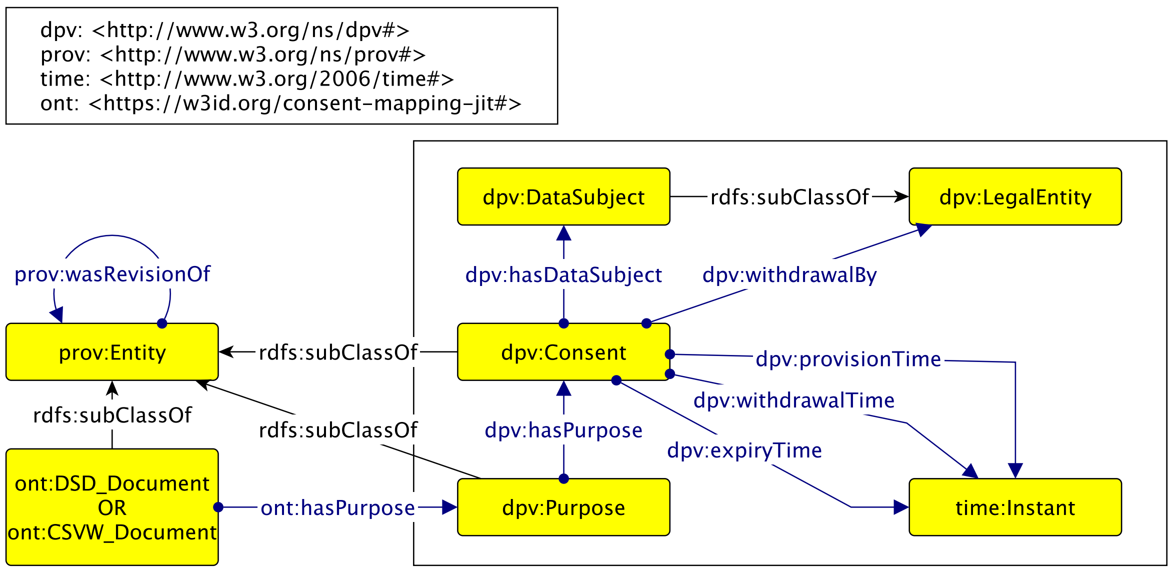


Fig. 2. Main concepts and interrelations of our model.

Although DPV provides most of the classes and relations we need (i.e., around Data Subject, Consent, and Purpose), we needed to introduce the link between a dataset (schema) and a data processing purpose.

In DPV, the class "Consent" represents the consent given by a Data Subject (in most cases) or their legal representative. In this, each consent is considered unique for a given combination of Data Subject, Purposes, Processing operations, and Personal Data. DPV provides attributes to model how and when the consent that has been given and subsequently withdrawn. In our scenario, we also wish to model the situation where a Data Subject chooses to not give their consent, which is something that DPV cannot currently represent. Additionally, the DPV also does not provide information regarding modeling of information regarding the evolution of consent over time – such as a Data Subject providing consent that has been expired, was previously withdrawn, or was refused. To model this information, we introduced PROV-O [15] into our consent ontology which allowed us to capture revisions of consent in the form of provenance of entities. DPV provides constructs for annotating consent instances with the dates and times they were given (`dpv:provisionTime`) and the time they were withdrawn

[4] ObjectPropertyRange(dpv:hasPurpose ObjectUnionOf(dpv:Consent dpv:PersonalDataHandeling))

(`dpv:withdrawalTime`). Temporal dimensions in DPV are captured using the Time Ontology[5]. We also avail of `dpv:expiryTime` to capture the date on which a data subject's consent expires. The class "Purpose" is used to represent the purpose of processing (personal) data, which may range from using email addresses to send a newsletter to using one's profile and purchase history for targeted advertising.

DPV only aims to provide a vocabulary and therefore does not prescribe methods for detecting erroneous or incomplete consent information. Rather than availing of Web Ontology Language (OWL) reasoning and the challenges that come with the Open World assumption, we validate our knowledge base with SHACL (Shapes Constraint Language) [18] which is the W3C Recommendation for prescribing conditions called 'shapes' which valid RDF graphs in a knowledge base must adhere to. SHACL is especially useful for closed-world assumptions and cardinality constraints which are the key constraints in our conceptual model. To summarize, we check whether there are no issues in the knowledge base at the graph level (i.e., RDF data model) rather than the semantic level (i.e., OWL). Another advantage of SHACL is the generation of machine-readable test results that can be utilized for documentation and compliance purposes.

4.1 Consent Ontology: Related and Prior Work

The work presented in [2] is, in short, focused on adopting Semantic Web technologies and extending standardized vocabularies for 1) prescribing the actions that should take place as well as the interactions between various stakeholders in the context of GDPR, and 2) analyze these models with respect to annotated logs and questionnaires to facilitate compliance verification. The work presented in [2] is from a prescriptive level concerned with the datasets that data processing activities use, who and how consent was obtained, and whether data processing activities comply with the consent that is and will be collected. While there is an overlap in terminology and the application domain, the model we propose in this article is meant to allow agents to utilize consent information available to an organization for assisting in the processing based on that consent and is not concerned with the assessment of the validity of consent for compliance. The model we propose in this article is intended to assist an organization in following the obligations of GDPR by generating a dataset taking into account the permission provided by given consent. Though there are some nuances in the interpretation of some terminology used in both studies, they are complementary in a broader narrative.

SPIRIT [7], SPECIAL [8] and the work presented in [19] have concepts related to data subjects, purposes, and data processing activities that are relevant. Those vocabularies were, however, were mostly developed for a posteriori compliance analysis of logs or assessing operations before their execution. In SPECIAL, their vocabulary is furthermore used to analyze whether a data controller's policy complies with a data subject's consent [20]. The SPECIAL compliance checker is used at an individual level for every data subject's consent. Our goal is to produce an entire dataset for a given collection of consent. The SPECIAL compliance checker also uses OWL2 subsumptions to dynamically check whether a given purpose or processing is compatible with a given consent which, while being more flexible, is rarely a dynamically changing operation as organizations have purposes that do not change frequently. While the intended uses of these vocabularies are different, their integration with the vocabularies we have introduced in this section will be challenging due to semantic heterogeneity of terms involved. An alignment of these–which would result in ontology mappings facilitating interoperability is considered for future work, though we note that the core vocabulary developed in the SPECIAL project informed DPV's base vocabulary.

Most relevant to the Consent Ontology is GConsent, which we proposed in [6]. GConsent recognizes that most initiatives do not consider all aspects of consent in order to appropriately assess whether consent fulfils the requirements provided by GDPR. GConsent therefore aims to capture the "context" of consent along with its evo-

[5] https://www.w3.org/TR/owl-time/, last accessed January 2020

lution as an entity based on the notion of distinct states such as not given to given to withdrawn. GConsent represents verbose information about consent from the perspective of compliance requirements rather than an organization's perspective. Consent Receipt [20]. is a standard by the non-profit Kantara Initiative that specifies attributes for capturing a consent record in the form of the human-readable receipts primarily intended for data subjects. The current iteration of Consent Receipt (v1.1) is based on the ISO/IEC 29100:2011 terminology, which is different from that used by the GDPR.

5 Generating Uplift Mappings in R2RML

To exemplify our approach, we will introduce a running example.[6] In our example, we simulate an organization who aims to send a newsletter to their customers (a data processing purpose). Newsletters can only be sent to customers who have given their explicit consent. In our example, we have a relational database table "Customer" that contains last names, first names, and email addresses (amongst other things).

Table 1. The customers table.

id	email	first_name	last_name	…
1	user_1@example.org	Firstname 1	Lastname 1	
2	user_2@example.org	Firstname 2	Lastname 2	
3	user_3@example.org	Firstname 3	Lastname 3	
…	…	…	…	…

We will use that table to a dataset schema file referencing that information. We assume, for the running example, that the dataset schema file will be stored at a particular location[7], this will be important as the location of the file (an absolute URI) will be the base of the RDF graph. When creating a dataset for data processing purposes, a location (and thus, also, an absolute URI) will be generated

5.1 Step 1: Annotating the Schema.

We start off with the (re)use of a schema describing the tabular data, which we will annotate with information on where to fetch the data. Listing 1 depicts an RDF graph containing a minimal dataset definition of the tabular data using the CSVW vocabulary and JSON-LD representation provided by [10]. We only retained the name of the columns as well as the property URLs giving us an indication as to how values in these columns must be interpreted. The vocabulary allows one to prescribe mandatory constraints (i.e., are values required), data types, and patterns that values should comply with (amongst others). These constraints are useful for validating tabular data. The highlighted statements extend the tabular file definitions with mapping information. We refer to a relational database table called "Customer" with the structure depicted in Table 2.

Table 2. Schema of the relational database table "Customers"

Column	Type	Null	Key
id	bigint(20)	No	primary
first_name	varchar(50)	No	
last_name	varchar(50)	No	
email	varchar(255)	No	Unique
…	…	…	…

[6] All files are available at https://chrdebru.github.io/papers/2020-kais/index.html. Note to reviewers: we will provide the files on a long-term storage platform with a suitable license upon acceptance/publication.

[7] E.g., http://www.example.org/data.csv

```
{
  "@id": "http://example.org/schema/mydataset.json",
  "@context": "http://www.w3.org/ns/csvw",
  "columns": [{
    "name": "first_name",
    "propertyUrl": "http://xmlns.com/foaf/0.1/givenName"
  }, {
    "name": "last_name",
    "propertyUrl": "http://xmlns.com/foaf/0.1/familyName"
  }, {
    "name": "email",
    "propertyUrl": "http://xmlns.com/foaf/0.1/mbox"
  }]
}
```

Listing 1 Definition of a tabular file.

We reuse R2RML's predicates to further annotate the dataset schema, as R2RML already provides us with the necessary predicates to annotate the file's schema. Those predicates are used to indicate where one can find the source tabular information in a relational database (see Listing 2, green highlights). It is important to note that we do not intend to create a valid R2RML document by reusing those predicates. We will, however, use them to generate a valid R2RML mapping in the next step of the process.

```
{
  "@id": "http://example.org/schema/mydataset.json",
  "@context": "http://www.w3.org/ns/csvw",
  "rr:tableName": "Customer",
  "columns": [{
    "name": "first_name",
    "propertyUrl": "http://xmlns.com/foaf/0.1/givenName",
    "rr:column": "first_name"
  }, {
    "name": "last_name",
    "propertyUrl": "http://xmlns.com/foaf/0.1/familyName",
    "rr:column": "last_name"
  }, {
    "name": "email",
    "propertyUrl": "http://xmlns.com/foaf/0.1/mbox",
    "rr:template": "mailto:{email}"
  }]
}
```

Listing 2 Extending the definition of a tabular files with mapping information.

The dataset schema is also serialized in JSON-LD [22], a common and increasingly popular representation for RDF for the available tooling. One can translate this JSON-LD into another RDF serialization such as TURTLE, but most of the column definitions will have no identifier (i.e., they will be blank nodes). Our approach of using RDF Data Cube had the additional benefit of having identifiers (URIs) for the different parts of the schema. This allows for an easier separation between schema and annotations. While this is also possible in JSON-LD, the way RDF Data Cube structures the data "forces" one to provide identifiers for those various components.

5.2 Step 2: 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 columns to create subject maps; 3) Create predicate object maps for the columns, and 4) connect the dataset that will be generated with its schema.

We obtain an executable R2RML mapping by merging the models resulting from each SPARQL CONSTRUCT query. This model is not meant to be merged with the prior RDF graphs from Listing 2. Instead, it will be used to generate RDF that will be the “just-in-time” dataset. 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. Note that we have omitted prefixes and base declarations from each of the query listings for brevity.

The generation of a logical table (tables and views) for each schema related to a table is the first query and is shown in Listing 3. A similar CONSTRUCT query is used for schemas related to a query with the `rr:query` predicate. The namespace `ont:` refers to our vocabulary developed for this study, and is useful in order to attach the different components of the R2RML mapping later on.

```
CONSTRUCT {
  [] rr:logicalTable [ rr:tableName ?t ] ;
     ont:correspondsWith ?x .
} WHERE {
  ?x rr:tableName ?t .
}
```

Listing 3 Generating an R2RML triples map for each schema.

The CONSTRUCT query for generating the subject map is shown in Listing 4. The columns of the tabular data are used to identify each record. We use that information to generate the subject map of a triples map. The table columns used by the columns are used for creating a template that will identify each record in the dataset. An R2RML processor will use the template, which will generate values, to keep information of each record in an appropriate data structure (e.g., a dictionary). Notice on line 17 we rely on `strSplit` function offered by Apache Jena’s SPARQL processor[8] to split a string into a list. Such a function was, unfortunately, not available in SPARQL. On lines 20 and 21 (gray) we include function calls which are part of an R2RML-F, an extension of R2RML [23].

```
 1. CONSTRUCT {
 2.   ?tm rr:subjectMap [
 3.     rr:termType rr:BlankNode .
 4.     rr:template ?x
 5.   ] .
 6. } WHERE {
 7.   ?tm rr:logicalTable [ rr:tableName ?t ] .
 8.   {
 9.     SELECT (concat("{", GROUP_CONCAT(?c ; separator='}-{'), "}") AS ?x)
10.     WHERE
11.     {
12.       { [ rr:column ?c ] }
13.       UNION
14.       { [ rr:template ?template ]
15.         BIND(strafter(?template, "{") AS ?tempx)
16.         BIND(replace(?tempx, "\\}[^\\}]*$", "") AS ?tempy)
17.         ?c apf:strSplit ( ?tempy "\\}[^\\{]*\\{" )
18.       }
19.       UNION
20.       { [ rrf:functionCall [
21.           rrf:parameterBindings ( [ rr:column ?c ] ) ] ] }
22.     }
```

[8] https://jena.apache.org/, last accessed March 2019

```
  }
}
```

Listing 4 Generating a subject map linking for records of the datasets.

Listing 5 provides the CONSTRUCT query for adding predicate object maps to the triples maps based on columns. The query for tabular data is fairly straightforward. We test for the presence of a `csvw:propertyURL` that refers to an RDF predicate. As the use of `csvw:propertyURL` is not mandatory, we will construct a predicate based on the column's name from the schema when that property is missing (lines 17 to 19). We note that a base URI –usually the URL or IRI of the dataset– is needed to provide absolute URIs of the column names. We avail of URI encoding to ensure that the URI of each predicate is valid.

```
 1. BASE <http://www.example.org/data.csv>
 2. # PREFIXES OMMITTED
 3. CONSTRUCT
 4. {
 5.   ?tm rr:predicateObjectMap [
 6.     rr:predicate ?pred ;
 7.     rr:objectMap [
 8.       ?p ?c .
 9.       rr:termType ?type .
10.     ]
11.   ]
12. }
13. WHERE
14. {
15.   ?tm ont:correspondsWith ?schema .
16.   ?x csvw:name ?name
17.   BIND(iri(concat("#", encode_for_uri(?name))) AS ?pred1)
18.   OPTIONAL { ?x csvw:propertyUrl ?temp . BIND(IRI(STR(?temp)) AS ?pred2) }
19.   BIND(IF(BOUND(?pred2), ?pred2, ?pred1) AS ?pred)
20.   ?x ?p ?c
21.   FILTER ( ?p IN ( rr:column, rr:template,
22.                    rr:constant, rrf:functionCall ) )
23.   OPTIONAL { ?x rr:termType ?type }
24. }
```

Listing 5 Generating predicate object maps from columns.

Finally, the dataset that will be generated with this dataset also needs to be connected to its schema. This is straightforward with the following CONSTRUCT query (in Listing 6). In CSVW, the schema refers to a particular file with the `csvw:url` predicate. We add this statement in the dataset that we will generate by executing the R2RML mapping as one of the final steps.

```
CONSTRUCT
{ ?schema csvw:url <http://www.example.org/data.csv> . }
WHERE
{ [ ont:correspondsWith  ?schema ] }
```

Listing 6 "Linking" the generated dataset with its schema.

With these mappings -- which are declarative and implemented as SPARQL CONSTRUCT queries -- we are able to generate an executable R2RML mapping that will be used in the generation of the "just-in-time" compliant dataset in the next process. Given our table "Customer" in **Table 2** and the snippets from Listing 2, the R2RML in Listing 7 is generated. While it is not explicit that the resource is a `rr:TriplesMap`, it will be inferred by the R2RML engine as such since the domain of `rr:logicalTable` is `rr:TriplesMap`.

```
[ ont:correspondsWith    <http://example.org/schema/mydataset.json> ;
  rr:logicalTable        [ rr:tableName  "Customer" ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [ rr:column  "last_name" ] ;
    rr:predicate  <http://xmlns.com/foaf/0.1/familyName>
  ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [ rr:template  " mailto:{email}" ] ;
    rr:predicate  <http://xmlns.com/foaf/0.1/mbox>
  ] ;
  rr:predicateObjectMap  [
    rr:objectMap  [ rr:column  "first_name" ] ;
    rr:predicate  <http://xmlns.com/foaf/0.1/givenName>
  ] ;
  rr:subjectMap          [
    rr:template  "{last_name}-{email}-{first_name}" ;
    rr:termType  rr:BlankNode
  ]
] .
```

Listing 7 Generated R2RML Mapping.

5.3 Step 3: Execution of the R2RML mapping

For the execution of our mapping, we rely on an implementation of the R2RML implementation developed by [23] as some use cases are not supported by the R2RML specification (see discussions). The mapping in Listing 7 contains no statements that fall outside R2RML's scope and should work with other implementations of the specification. The execution of this mapping generated 3 RDF triples for each record in the Customers table, which corresponds with the generated dataset in **Fig. 1**. An example of such a record is shown in Listing 8, representing "user_1" in the graph with their first and last name.

```
[ <http://xmlns.com/foaf/0.1/familyName>     "Lastname 1" ;
  <http://xmlns.com/foaf/0.1/givenName>      "Firstname 1" ;
  <http://xmlns.com/foaf/0.1/mbox>           <mailto:user_1@example.org>
] .
```

Listing 8 A record generated with the R2RML mapping of Listing 7

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 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 "Natural Mapping of SQL Values" of [16].

5.4 Step 4: Validating the generated RDF

We validate the generated RDF by checking the integrity constraints described in the schema. This is necessary as execution of any R2RML mapping according to a particular schema or standard, does not guarantee does not guarantee that the resulting dataset complies with the constraints of that schema or standard. For RDF Data Cubes, the specification presents a set of so-called integrity constraints in the specification [13], which are a sequence of SPARQL

ASK queries. For CSVW, we rely on CSVW validators[9] taking as input both the schema and the generated dataset to check whether the dataset adheres to the schema's constraint.

5.5 Related Work on Generating Mappings

Though tools exist to convert between representations (such as from OLAP and CSV to RDF Data Cube [24]), we are interested in the state of the art of generating mappings from one representation to another. Related work in generating (R2RML) mappings from other representations is quite limited. The authors in [25] – 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.

The Open Cube Toolkit [26] provides a D2RQ [27] 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 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 relate to their mapping language; it is bespoke, not in RDF and has not been declared in a particular namespace.

6 Creating Compliant Datasets

In this section, we will demonstrate our approach by 1) describing how we use a knowledge base using the ontology we presented in the previous section; 2) annotate the schemas for generating R2RML mappings (cf. Section 3.1); and 3) how we combine 1) and 2) to generate a consent-compliant dataset for use in data processing operations.

6.1 Using the Consent Ontology

We retrieve consent information for a particular purpose using a SPARQL SELECT query (see Listing 9) upon the Consent Information Knowledge Base (whose information is structured according to the Consent Ontology). We ensure retrieving the most recent consent information for each user by removing those for which there is an instance of consent with a more recent date (lines 12-17). We then filter out those consent instances which have been withdrawn and are not expired (lines 19-24)). Note, on line 19, that the consent can be withdrawn by a resource other than the data subject. While it is usually the data subject that gave their consent, there are cases where consent can be withdrawn (or given) by representatives (e.g., a parent for their child).

```
 1. PREFIX dcterms: <http://purl.org/dc/terms/>
 2. PREFIX dpv: <http://www.w3.org/ns/dpv#>
 3. PREFIX prov: <http://www.w3.org/ns/prov#>
 4. PREFIX time: <http://www.w3.org/2006/time#>
 5.
 6. SELECT ?consent ?user WHERE {
 7.   # OBTAIN ALL LATEST CONSENT (INFORMATION)
 8.   ?consent dpv:hasDataSubject ?user .
 9.   ?consent dpv:hasPurpose <http://data.example.com/purpose/30> .
10.   ?consent dpv:provisionTime [ time:inXSDDateTime ?datetime ] .
11.
12.   FILTER NOT EXISTS {
13.     [ dpv:hasPurpose <http://data.example.com/purpose/30> ;
14.       dpv:hasDataSubject ?user ;
```

[9] E.g., https://github.com/malyvoj3/csvw-validator

```
      dpv:provisionTime [ time:inXSDDateTime ?datetime2 ] ]
      FILTER(?datetime2 > ?datetime)
  }

  # RETAIN THOSE NOT WITHDRAWN AND STILL VALID
  FILTER NOT EXISTS { ?consent dpv:withdrawalBy [] }
  ?consent dpv:expiryTime [
    time:inXSDDateTime ?expirytime
  ]
  FILTER(NOW() < ?expirytime)
}
```

Listing 9 Retrieving the latest and valid consent information for a specific data processing purpose.

In Section 3, we annotated the schema with references to tables, columns, and so on. Similarly, we annotate the schema with references to a purpose. To exemplify this process, we will expand the running example for which we created a synthetic dataset. In that dataset, we have simulated ten users engaging with the terms and conditions of five systems. The terms and conditions of a system can evolve over time and, when they change, the consent has to be obtained again. Users can either reject, give, or withdraw their consent for a particular purpose at any given time. In this dataset,

- the policies of systems #4 and #5 were updated twice, and that of system #2 only once;
- there are 29 instances of users renewing consent;
- there are 68 instances of users re-giving consent after withdrawing; and
- users withdrew their consent 102 times.

For the synthetic dataset, we have a simple schema (see Listing 10). We see in this listing that this schema retrieves data from a table and that the values for the dimension and measure are obtained from columns. Those values are used to create IRIs, one to identify records and one for mailboxes (according to the FOAF vocabulary). We note that the schema is annotated for a particular purpose (line 9)—an URI referring to sending newsletters. For the sake of brevity, however, we have omitted the first- and last names from the annotated schema.

```
 1. {
 2.   "@id": "http://example.org/schema/mydataset.json",
 3.   "@context": {
 4.      "@vocab": "http://www.w3.org/ns/csvw#",
 5.      "ont": "https://w3id.org/consent-mapping-jit#",
 6.      "rr": "http://www.w3.org/ns/r2rml#"
 7.   },
 8.   "rr:tableName": "Customers",
 9.   "ont:forPurpose": { "@id": "http://data.example.com/purpose/30" },
10.
11.   "columns": [{
12.     "name": "user",
13.     "propertyUrl": "http://purl.org/dc/terms/identifier"},
14.     "rr:template": "http://data.example.com/user/{id}"}
15.   }, {
16.     "name": "email",
17.     "propertyUrl": "http://xmlns.com/foaf/0.1/mbox",
18.     "rr:template": "mailto:{email}"}
19.   }]
20. }
```

Listing 10 A schema where each record is identified by a user and the value is the email address.

In RDF Data Cube, we must explicitly identify the components that identify an observation (corresponds with "records"). In CSVW, however, this is not mandatory. We, therefore, use `dct:identifier` for indicating which column should be interpreted as an identifier (line 13 in Listing 10). The reason being that CSVW prescribes the use of predicates whose node names reside in the base name of the file. For example, the column with name "name" will become the named node name in the file with URI `http://example.org/schema/mydataset.json`. The R2RML mapping that is the result of executing a sequence of the mapping generation `CONSTRUCT queries` is shown in Listing 11.

```
[
  ont:correspondsWith   <http://example.org/schema/mydataset.json> ;
  rr:logicalTable       [ rr:tableName  "user" ] ;
  rr:predicateObjectMap [
   rr:objectMap [ rr:template  "http://data.example.com/user/{id}" ] ;
   rr:predicate  <http://purl.org/dc/terms/identifier>
  ] ;
  rr:predicateObjectMap [
    rr:objectMap  [ rr:template  "mailto:{email}" ] ;
    rr:predicate  <http://xmlns.com/foaf/0.1/mbox>
  ] ;
  rr:subjectMap [
    rr:template  "{id}-{email}" ;
    rr:termType  rr:BlankNode
  ]
] .
```

Listing 11 Generated R2RML Mapping (II).

6.2 Creating a Compliant Dataset

The synthetic dataset contains, for purpose #30 "sending newsletters", consent information about three users. These users have given and withdrawn their consent several times in the 100 iterations. In the end, user #10 has accepted the purpose, whereas users #8 and #9 have withdrawn their consent. The execution of the R2RML mapping (Listing 11) results in a dataset that contains all users, but for our approach to work we have to filter out only those for which we have explicit consent, in order to generate the target Compliant Dataset (as per Figure 1). While we could manipulate the source of the R2RML mapping (by creating an SQL query with necessary WHERE clauses) to do the filtering, we choose to use SPARQL instead. This approach allows us not only to cache intermediate outputs but also allows us to avail of R2RML dialects that support non-relational data such as RML [28] and xR2RML [29]. In other words, we become less dependent on underlying database technologies that may differ across (different parts of) organizations and rely on RDF and SPARQL as the common model.

With the query in Listing 9, we fetch the latest consent information of users. This information is used to construct a list of users that have given their consent. That list is then used to create and execute a SPARQL DESCRIBE query returning to us a dataset that is fit for the purpose for which consent has been agreed. The list is used to create a VALUES clause. The DESCRIBE query is shown in Listing 12 and is applied to the dataset to obtain a subset with only the information of those that have given their consent.

```
DESCRIBE ?record WHERE {
  ?record  <http://purl.org/dc/terms/identifier>  ?dim
  VALUES ?dim { <http://data.example.com/user/10> }
}
```

Listing 12 Obtaining a subset of the generated dataset that contains only the information of those who have given consent.

The predicate highlighted in green in Listing 12 corresponds with the predicate of the identifier used in Listing 10. We inspect the schema to obtain those when creating the DESCRIBE query. We omit this rather simple SPARQL query.

We already stated that for this particular purpose, user #10 has given their consent, and users #8 and #9 have withdrawn their consent. Because of this, only user #10 is retained in the result, Listing 13 shown below. If multiple records were retained, then the VALUES clause in Listing 12 would have contained multiple IRIs separated by spaces. This dataset is validated with the same techniques as Section 5.4.

```
[ <http://example.org/schema/mydataset.json#email>
          <mailto:user_10@example.org> ;
  <http://example.org/schema/mydataset.json#user>
          <http://data.example.com/user/10>
] .
```

Listing 13 Resulting compliant dataset after executing the SPARQL DESCRIBE query of Listing 12. Note that the URI of the dataset corresponds with that of the constant in Listing 11.

6.3 Provenance Information

Capturing provenance information is not a separate step, but rather a process that is happening throughout the steps of generating an R2RML mapping and compliant dataset. Provenance information provides insights on a resource's origin, such as who created that resource, when it was modified, or how it was created [30]. PROV-O [15], which we already adopted for the consent ontology of this study, is a W3C Recommendation for representing and exchanging provenance information as RDF. PROV-O's core concepts and relations 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 `ont:R2RML_Processor` and `ont:Mapping_Generator` will be instantiated for our implementation.

Table 3. Extending PROV-O

Class	Parent Class	Description
`ont:DSD_Document` `ont:CSVW_Document`	`prov:Entity`	Used to represent RDF graphs containing our annotated schemas
`ont:R2RML_Mapping`	`prov:Entity`	Used to represent the generated R2RML mappings
`ont:Dataset`	`prov:Entity`	Used to represent the generated datasets
`ont:Compliant_Dataset`	`prov:Entity`	Used to represent the final datasets
`ont:Validation_Report`	`prov:Entity`	Used to represent the validation reports
`ont:Generate_Mapping`	`prov:Activity`	Represents the activity of generating an R2RML mapping from an annotated schema
`ont:Execute_Mapping`	`prov:Activity`	Represents the activity of executing the R2RML mapping
`ont:Filter_Dataset`	`prov:Activity`	Represents the activity of filtering the dataset for creating a compliant dataset
`ont:Validate_Dataset`	`prov:Activity`	Represents the activity of validating a dataset
`ont:Mapping_Generator`	`prov:SoftwareAgent`	Represents the software agent that generates an R2RML mapping as per our approach
`ont:R2RML_Processor`	`prov:SoftwareAgent`	Represents the software agent executing the mapping
`ont:Validator`	`prov:SoftwareAgent`	Represents the software agent validating the dataset

We declare relations between instances of our subclasses to capture the whole process. For instance, a `ont:Generate_Mapping` uses (`prov:uses`) a `ont:CSVW_Document` to generate a `ont:R2RML_Mapping`. A mapping is thus generated by (`prov:wasGeneratedBy`) an activity. This activity was performed (`prov:wasAssociatedWith`) by our implementation of our approach (`ont:Mapping_Generator`). The mapping is also derived from the `ont:CSVW_Document`, so we also assert a `prov:wasDerivedFrom` between the two entities. 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 schema can be used to generate an executable R2RML document multiple times. This helps us fulfill some of the requirements put forward by policies.

As for related work on creating compliant datasets, there is little relate work, To the best of our knowledge, relevant academic work on generating compliant datasets "just-in-time" (i.e., on-demand) is limited to [9].

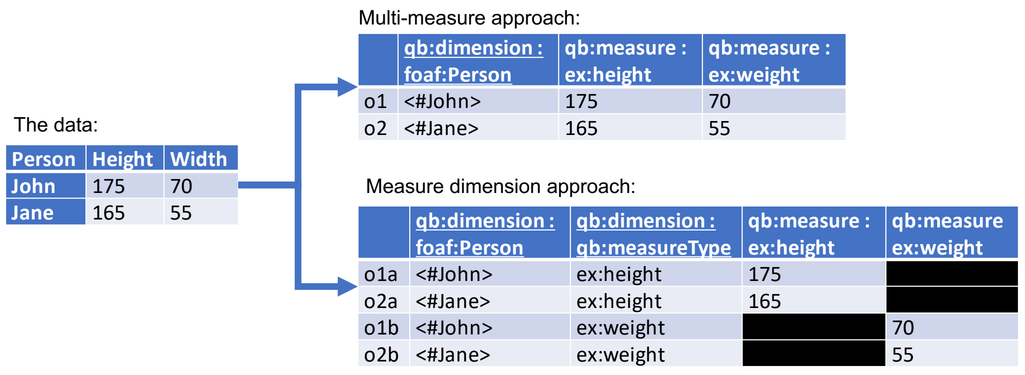
7 On Generalizing the Approach

In this section, we provide the motivation for supporting tabular representations next to RDF Data Cubes as well as the differences in the implementation (i.e., the SPARQL queries). This allows one to identify which parts of our approach need to be addressed if one were to provide support for another representation.

7.1 Motivating the Adoption of CSVW

Prior work [12] focused on the generation of RDF Data Cube Vocabulary [13] datasets. The Data Cube Vocabulary allows one to represent multi-dimensional datasets using RDF. 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. As it is capable of representing a wide variety of datasets and is, unlike other vocabularies, standardized, it was deemed suitable for our purpose. However, we realized that RDF Data Cubes can become too complex for certain data processing tasks.

The vocabulary works well when observations have at most one measured value. Problems may arise when one needs to capture multiple observations (e.g., rainfall and temperature, or width and height). The Recommendation proposes two approaches: a measure dimension approach and a multi-measure approach [13]. Take for instance the example on the left-hand side of **Fig. 3**. Here we have a rather simplistic representation of two observations. The two observations o1 and o2 are identified by instances of people (dimension). For each person, we record their height and weight.



Person	Height	Width
John	175	70
Jane	165	55

	qb:dimension : foaf:Person	qb:measure : ex:height	qb:measure : ex:weight
o1	<#John>	175	70
o2	<#Jane>	165	55

	qb:dimension : foaf:Person	qb:dimension : qb:measureType	qb:measure : ex:height	qb:measure ex:weight
o1a	<#John>	ex:height	175	■
o2a	<#Jane>	ex:height	165	■
o1b	<#John>	ex:weight	■	70
o2b	<#Jane>	ex:weight	■	55

Fig. 3. Left: Height and width measured for two people. The measurement units of height and width were omitted but are assumed to be centimeters and kilograms. Right: The multi-measure approach for the example on the left and the measure dimension approach for the example for that same example. Note that we have omitted namespaces, but assume to be using URIs for instances of `foaf:Person` and predicates for weight and height from an arbitrary namespace. All predicates with the prefix `qb` reside in RDF Data Cube's schema.

The multi-measure approach resembles the "relational" approach the closest but comes at an important price: expressiveness. With the Data Cube Vocabulary, one can only declare one unit of measurement per observation. Those can be declared as an attribute of the predicates, but then those are declared for the whole (and even all) datasets and you cannot use nits interchangeably (e.g, `dbpedia:Centimetre` and `dbpedia:Metre`).

In the measure dimension approach, observations are said to be identified by an additional dimension: the type of the measure. The recommendation prescribes the use of `qb:measureType` to provide that additional dimension. The URI for the observed value does appear twice in each observation: once as the object for the `qb:measure-Type` property, and once as the predicate for the actual observed value. With this approach, units can now be attached for each observed value; e.g., `dbpedia:Kilogram` for weight and `dbpedia:Centimetre` for height. But one then has n observations per "row", with n the number of measured values, creating a complex graph.

Users are thus faced with a rather difficult decision; limited "relational" data or expressive complex graphs. With these limitations in mind, we have chosen to adopt a tabular representation allowing for the annotation of tables and columns (datatype, cardinality, etc.) [10].

7.2 Differences Between RDF Data Cubes and CSVW

This article has demonstrated how an initial approach to generating compliant datasets can be generalized to cater for different types of data; tabular data using an RDF representation of CSV files (which can be translated into CSV files according to the CSVW specification), and RDF Data Cubes based on the aforementioned prior work.

This article focuses on CSV files. Not only are these representations useful for a wider community (e.g., via the tooling available for processing JSON and CSV files), but it meant the instantiation of our approach was simpler as well.

Below we elaborate on the differences in terms of the "implementation" of the approach. Note that these are limited to the number of SPARQL CONSTRUCT queries that were required. In other words, the design of the approach is – from a conceptual perspective – the same, but RDF Data Cubes required additional transformations for the creation of a compliant dataset. The following are the key differences:

1. When dealing with RDF Data Cubes, we annotate the Data Structure Definitions. Those DSDs can be used for multiple datasets. When generating an R2RML mapping, we create a mapping for the generation of observations. Those observations have to be related to an instance of a dataset, which in turn is related to the DSD. When creating the subject map for RDF Data Cubes, we also: 1) include a predicate object map referring to an instance of a dataset (a constant), and 2) create a URI for that dataset based on the URI base.

2. For CSVW, each record is assumed to be identified by all the values. In RDF Data Cube, each observation is identified by its dimensions. The SPARQL CONSTRUCT query of Listing 4 will only consider column references used in dimensions.

3. The predicate object maps for CSVW are straightforward; a literal for each value. In RDF Data Cubes, there are dimensions and measures. We proposed a SPARQL CONSTRUCT query for each that resembles Listing 5. Key differences with respect to CSVW is that both can have optional term types (e.g., a resource or a literal) and ranges (e.g., XSD datatypes).

4. As for validating the datasets, RDF Data Cube prescribes a set of SPARQL ASK queries implementing so-called “integrity constraints” that test whether a dataset adheres to a Data Structure Definition. These SPARQL queries have been implemented in our service. For CSVW, the standard only prescribes what types of validations are supported by the specification. The community group behind this Recommendation also

provided 282 test cases that validators should cover.[11] Rather than implementing a validator, we availed of one realized with an MIT license.[12]

We refer to [12] for more details on the implementation for RDF Data Cube.

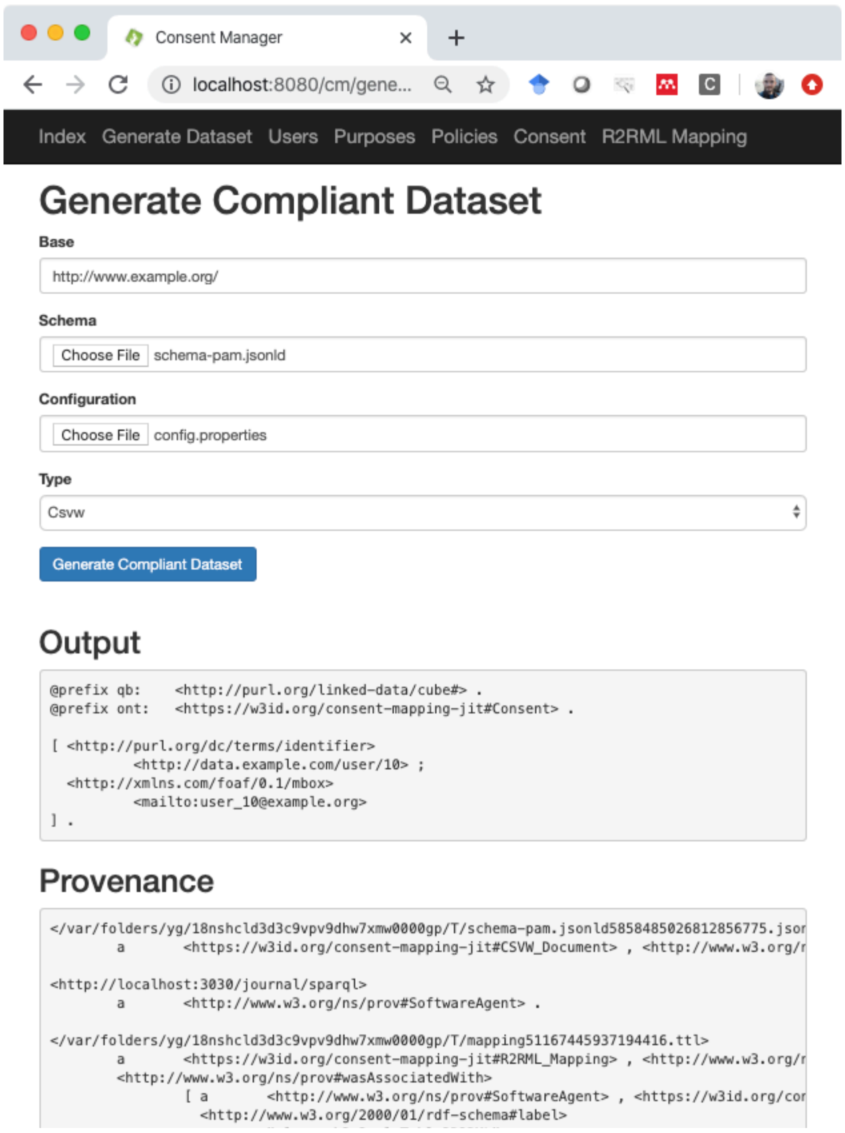
8 Design of the Service

We implemented our approach as a service. The service is implemented in Java EE and uses the Apache Jena[13] to process the RDF. External libraries include R2RML-F[14] as the R2RML processor and a CSVW validator which we referred to in the previous section. We also built an interface on top of that service (see **Fig. 4**) as a demonstrator, which was implemented with Apache Tapestry[15]. First, we assume that one maps the consent information stored in some non-RDF format to our consent knowledge base (by using, for example, R2RML) and expose that information via a SPARQL endpoint. This assumption is motivated by the fact that organizations oftentimes store information in various (non-RDF) data sources. Both the service and the SPARQL endpoint are residing on a server and that the endpoint is not accessible from outside the system (i.e., behind a firewall). This is usually the case for any database in an organizational setting.

To use the service, one needs to input an annotated schema and a link to a purpose to obtain the filtered dataset for a particular data processing activity. In **Fig. 4**, the annotated schema includes links to a purpose in one file (see Listing 10). The tool also requires information about the non-RDF datastore (location or connection URL, username and password if needed, etc.). Where this configuration should be stored is not detailed by any of the W3C Recommendations but depends on the various implementations. R2RML-F, the R2RML processor we have adopted, relies on a properties file. Our approach can be easily extended to store such information in separate graphs and instead point to the URI of a datastore description in RDF. We note that any governance activities related to creating the schema as well as the use of the obtained dataset are outside the scope of this paper.

Fig. 4 furthermore depicts the results of running the service: we have the resulting dataset and the provenance graph. All (intermediate) files are to be stored in a secure location with references (URLs) in the provenance graph. This allows one to verify all files generated by the activities for posthoc compliance checking. Where and how these files should be stored – which is again concerned with governance strategies – falls outside the scope of this article. Although an appropriate manager should for these governance strategies, we currently store these files in RDF files in a separate folder.

[11] https://w3c.github.io/csvw/tests/, last accessed January 2020
[12] https://github.com/malyvoj3/csvw-validator
[13] https://jena.apache.org/documentation/fuseki2/
[14] https://github.com/chrdebru/r2rml
[15] https://tapestry.apache.org/


Consent Manager
localhost:8080/cm/gene...
Index Generate Dataset Users Purposes Policies Consent R2RML Mapping

Generate Compliant Dataset

Base: http://www.example.org/

Schema: Choose File schema-pam.jsonld

Configuration: Choose File config.properties

Type: Csvw

Generate Compliant Dataset

Output

```
@prefix qb:    <http://purl.org/linked-data/cube#> .
@prefix ont:   <https://w3id.org/consent-mapping-jit#Consent> .

[ <http://purl.org/dc/terms/identifier>
          <http://data.example.com/user/10> ;
  <http://xmlns.com/foaf/0.1/mbox>
          <mailto:user_10@example.org>
] .
```

Provenance

```
</var/folders/yg/18nshcld3d3c9vpv9dhw7xmw0000gp/T/schema-pam.jsonld5858485026812856775.jsor
        a       <https://w3id.org/consent-mapping-jit#CSVW_Document> , <http://www.w3.org/r

<http://localhost:3030/journal/sparql>
        a       <http://www.w3.org/ns/prov#SoftwareAgent> .

</var/folders/yg/18nshcld3d3c9vpv9dhw7xmw0000gp/T/mapping5116744593719416.ttl>
        a       <https://w3id.org/consent-mapping-jit#R2RML_Mapping> , <http://www.w3.org/r
        <http://www.w3.org/ns/prov#wasAssociatedWith>
                [ a       <http://www.w3.org/ns/prov#SoftwareAgent> , <https://w3id.org/cor
                  <http://www.w3.org/2000/01/rdf-schema#label>
```

Fig. 4. Interface built on top of the service. Given an annotated schema and details of the data sources (in a configuration file), the service generates a dataset taking into account the consent an organization has gathered as well as a provenance graph with details of each step as outlined in Section 3.

Fig. 5 is generated using a tool called "ontology-visualization"[16], and depicts a part of the provenance graph generated by the tool. As the graph is quite large, we have omitted namespaces and only display the statements surrounding the activity of generating an R2RML mapping. This figure illustrates the detailed provenance information we can store. Again, the inclusion of additional information (such as the person executing the service) can easily be included but requires integration with data governance platforms.

[16] https://github.com/usc-isi-i2/ontology-visualization, last accessed May 20, 2019

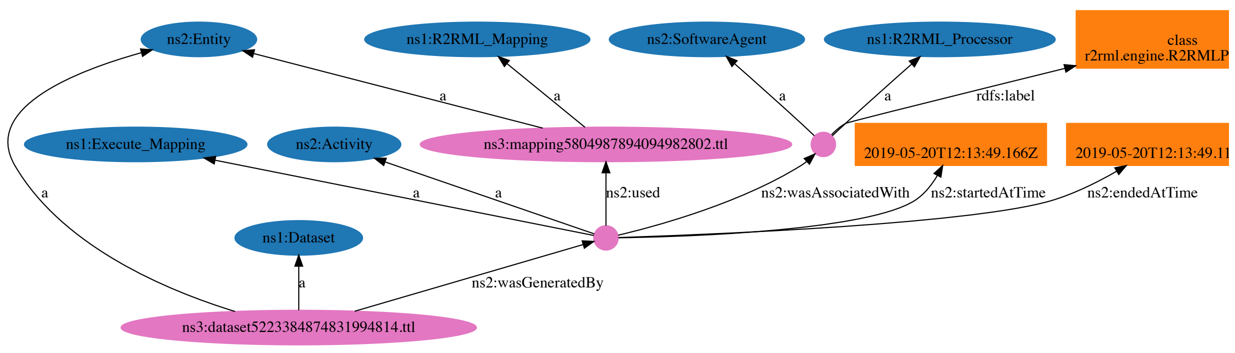


Fig. 5. Visualization of some provenance information generated by the service. We have omitted namespaces and only display a part of the provenance graph for brevity.

9 Conclusions and Future Work

We argue that datasets are used by an organization for a specific purpose and that datasets should be generated suitable for the intended purpose, including any organizational policies it should comply with. The question we addressed in this article is: "How can we generate datasets for a specific purpose "just-in-time" that complies with given consent?" We believe we answered this question by proposing a solution which:

1. Shows how we capture consent information by extending DPV, the Data Privacy Vocabulary which is on track for standardization. Since we had to relate dataset schemas with DPV, we had to include concepts for representing those schemas and the link between the schemas and `dpv:Purpose`. Our extension is used to access such information an organization has gathered for a specific purpose and for a specific schema. This corresponds with the first objective O1 of our research which we outlined in Section 2.

2. To achieve the second objective O2 "generating mappings", we propose a method for generating R2RML mappings from annotated schemas. Those R2RML mappings, linked to a particular purpose and data source, are then used to generate RDF. Executing these mappings result in datasets that comply with the schema. This work was based on [12], and now generalized to cope with different types of datasets.

3. The third objective O3 "generating compliant datasets" proposed a declarative approach to manipulating the resulting datasets to exclude data for which no consent is given. This work was based on [9], and also generalized to cope with different types of datasets.

4. Finally, we achieved our fourth objective O4 by integrating all components into a service.

We have thus demonstrated that the use of semantic technologies for creating given-consent compliant datasets is not only feasible but also facilitates compliance verification processes and provides organizations with a compliance-by-design approach for their operations. These processes are facilitated by the declarative approach (i.e., queries), which is transparent. The generated mappings, datasets, and provenance data, in addition, can also be safely stored for posthoc analysis.

The system creates datasets with the goal of facilitating compliance. While it does so automatically, our approach allows one to look up why someone's personal data was included in a particular dataset via the consent knowledge base, the sequence of SPARQL queries, and the timestamps captured in the provenance model.

In terms of future work, we identified in this article both the need for a fuller evaluation of the implemented system and also the need for alignment across existing consent related vocabularies.

We could not report as yet on the evaluation of our approach via quantitative methods or the demonstration of the approach in a testbed, as we have as yet no access to a real-world dataset of given consent. The state of the art also provides little information on the actual data available and the complexity of the use cases. Instead, we relied on the demonstrator of Section 3 to convey the viability of our approach. We also created a script that generated synthetic data to support said demonstrator.

The challenge in aligning the various vocabularies is combining their respective scopes as well as their varying degrees of granularity (a specific type of heterogeneity). The work presented in this article, for instance, is meant to support data processing activities from an organizational perspective whereas the work presented in GConsent [6] aims at representing the consent given from the perspective of the data subject. Where our model relates dataset schemas with consent, GConsent and DPV have more fine-grained representations for personal data and data categories. An attempt at aligning the vocabularies should take into consideration the scope and objective of two vocabularies and their respective roles in the compliance process? Such an alignment may furthermore have an impact on how people engage with the vocabularies (e.g., the complexity of queries).

References

1. Landwher, C.: 2018: A Big Year for Privacy. Commun. ACM. 62, 20–22 (2019). https://doi.org/10.1145/3300224.
2. Pandit, H.J., O'Sullivan, D., Lewis, D.: Queryable Provenance Metadata For GDPR Compliance. In: Proceedings of the 14th International Conference on Semantic Systems (SEMANTiCS 2018), Vienna, Austria, Sep. 10-13, 2018. pp. 262–268. Elsevier (2018). https://doi.org/10.1016/j.procs.2018.09.026.
3. Pan, J.Z., Vetere, G., Gómez-Pérez, J.M., Wu, H. eds: Exploiting Linked Data and Knowledge Graphs in Large Organisations. Springer (2017). https://doi.org/10.1007/978-3-319-45654-6.
4. Pandit, H.J., Fatema, K., O'Sullivan, D., Lewis, D.: GDPRtEXT - GDPR as a Linked Data Resource. In: Gangemi, A., Navigli, R., Vidal, M.-E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., and Alam, M. (eds.) The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings. pp. 481–495. Springer (2018). https://doi.org/10.1007/978-3-319-93417-4_31.
5. 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: CEUR Workshop Proceedings (2017).
6. Pandit, H.J., Debruyne, C., O'Sullivan, D., Lewis, D.: GConsent - A consent ontology based on the GDPR. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 270–282 (2019). https://doi.org/10.1007/978-3-030-21348-0_18.
7. Westphal, P., Fernández, J.D., Kirrane, S., Lehmann, J.: SPIRIT: A Semantic Transparency and Compliance Stack. In: Khalili, A. and Koutraki, M. (eds.) Proceedings of the Posters and Demos Track of the 14th International Conference on Semantic Systems co-located with the 14th International Conference on Semantic Systems (SEMANTiCS 2018), Vienna, Austria, September 10-13, 2018. CEUR-WS.org (2018).
8. Kirrane, S., Fernández, J.D., Dullaert, W., Milosevic, U., Polleres, A., Bonatti, P.A., Wenning, R., Drozd, O., Raschke, P.: A Scalable Consent, Transparency and Compliance Architecture. In: Gangemi, A., Gentile, A.L., Nuzzolese, A.G., Rudolph, S., Maleshkova, M., Paulheim, H., Pan, J.Z., and Alam, M. (eds.) The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers. pp. 131–136. Springer (2018). https://doi.org/10.1007/978-3-319-98192-5_25.
9. Debruyne, C., Pandit, H.J.J., Lewis, D., O'Sullivan, D.: Towards Generating Policy-Compliant Datasets. In: 13th IEEE International Conference on Semantic Computing, ICSC 2019, Newport Beach, CA, USA, January 30 - February 1, 2019. pp. 199–203. IEEE (2019). https://doi.org/10.1109/ICOSC.2019.8665631.
10. Tennison, J.: CSV on the Web: A Primer, https://www.w3.org/TR/tabular-data-primer/.
11. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. J. Data Semant. 1, 147–185 (2012). https://doi.org/10.1007/s13740-012-0008-7.
12. Debruyne, C., Lewis, D., O'Sullivan, D.: Generating Executable Mappings from RDF Data Cube Data Structure Definitions. In: On the Move to Meaningful Internet Systems: OTM 2018 Conferences - Confederated International Conferences: CoopIS, CT&C, and ODBASE 2018, Valletta, Malta, October 22-26, 2018. Proceedings. pp. 333–350. Springer (2018). https://doi.org/10.1007/978-3-030-02671-4_21.
13. Cyganiak, R., Reynolds, D.: The RDF Data Cube Vocabulary, https://www.w3.org/TR/vocab-data-cube/.
14. Pandit, H.J., Polleres, A., Bos, B., Brennan, R., Bruegger, B.P., Ekaputra, F.J., Fernández, J.D., Hamed, R.G., Kiesling, E., Lizar, M., Schlehahn, E., Steyskal, S., Wenning, R.: Creating a Vocabulary for Data Privacy - The First-Year Report of Data Privacy Vocabularies and Controls Community Group {(DPVCG)}. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., and Meersman, R. (eds.) On the Move to Meaningful Internet

Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings. pp. 714–730. Springer (2019). https://doi.org/10.1007/978-3-030-33246-4_44.

15. Lebo, T., McGuinness, D., Sahoo, S.: PROV-O: The PROV Ontology, https://www.w3.org/TR/prov-o/.
16. Das, S., Cyganiak, R., Sundara, S.: R2RML: RDB to RDF Mapping Language, https://www.w3.org/TR/r2rml/.
17. Falco, R., Gangemi, A., Peroni, S., Shotton, D.M., Vitali, F.: Modelling OWL Ontologies with Graffoo. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., and Tordai, A. (eds.) The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers. pp. 320–325. Springer (2014). https://doi.org/10.1007/978-3-319-11955-7_42.
18. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL), https://www.w3.org/TR/shacl/.
19. Lehmann, J., Sejdiu, G., Bühmann, L., Westphal, P., Stadler, C., Ermilov, I., Bin, S., Chakraborty, N., Saleem, M., Ngomo, A.-C.N., Jabeen, H.: Distributed Semantic Analytics Using the SANSA Stack. In: d'Amato, C., Fernández, M., Tamma, V.A.M., Lécué, F., Cudré-Mauroux, P., Sequeda, J.F., Lange, C., and Heflin, J. (eds.) The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II. pp. 147–155. Springer (2017). https://doi.org/10.1007/978-3-319-68204-4_15.
20. (WU), S.K., (CeRICT), P.B., (WU), J.D.F., (CeRICT), C.G., (CeRICT), L.S., (CeRICT), D.D., (CeRICT), I.P., (CeRICT), I.S.: D2.8 - Transparency and Compliance Algorithms V2. (2018). https://doi.org/10.5281/ZENODO.2543622.
21. Lizar, M., Hodder, M.: Usable consents tracking and managing use of personal data with a consent transaction receipt. In: UbiComp 2014 - Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 647–652. Association for Computing Machinery, Inc (2014). https://doi.org/10.1145/2638728.2641681.
22. Sporny, M., Longley, D., Kellog, G., Lanthaler, M., Lindstrom, N.: JSON-LD 1.1: A JSON-based Serialization for Linked Data. W3C Community Group Draft Report, https://json-ld.org/spec/latest/json-ld/.
23. Debruyne, C., O'Sullivan, D.: R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings. In: Auer, S., Berners-Lee, T., Bizer, C., and 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-WS.org (2016).
24. Salas, P.E.R., Martin, M., Mota, F.M. Da, Auer, S., Breitman, K., Casanova, M.A.: Publishing statistical data on the web. In: Proceedings - IEEE 6th International Conference on Semantic Computing, ICSC 2012 (2012). https://doi.org/10.1109/ICSC.2012.16.
25. Skjæveland, M.G., Giese, M., Hovland, D., Lian, E.H., Waaler, A.: Engineering ontology-based access to real-world data sources. J. Web Semant. (2015). https://doi.org/10.1016/j.websem.2015.03.002.
26. Kalampokis, E., Nikolov, A., Haase, P., Cyganiak, R., Stasiewicz, A., Karamanou, A., Zotou, M., Zeginis, D., Tambouris, E., Tarabanis, K.: Exploiting linked data cubes with OpenCube toolkit. In: CEUR Workshop Proceedings (2014).
27. Bizer, C., Seaborne, A.: D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs. In: Proceedings of the 3rd International Semantic Web Conference (ISWC2004) (2004). https://doi.org/10.1.1.126.2314.
28. Dimou, A., Sande, M. Vander, Colpaert, P., Verborgh, R., Mannens, E., de Walle, R. Van: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Bizer, C., Heath, T., Auer, S., and Berners-Lee, T. (eds.) Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. CEUR-WS.org (2014).
29. Michel, F., Djimenou, L., Faron-Zucker, C., 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 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015. pp. 443–454. SciTePress (2015). https://doi.org/10.5220/0005448304430454.
30. Zhao, J., Hartig, O.: Towards interoperable provenance publication on the linked data web. In: CEUR Workshop Proceedings (2012).
31. Bizer, C.: D2R MAP - A Database to RDF Mapping Language. In: Proceedings of the Twelfth International World Wide Web Conference - Posters, WWW 2003, Budapest, Hungary, May 20-24, 2003 (2003).