A survey of RDB to RDF translation approaches and tools

Franck Michel, Johan Montagnat, Catherine Faron Zucker

LABORATOIRE

INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR7271

A survey of RDB to RDF translation approaches and tools

Franck Michel, Johan Montagnat, Catherine Faron-Zucker

Equipes Modalis/Wimmics

Rapport de Recherche
ISRN I3S/RR 2013-04-FR
Version 2

May 2014 - 23 pages

Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) - UMR7271 - UNS CNRS
2000, route des Lucioles - Les Algorithmes - bât. Euclide B 06900 Sophia Antipolis - France
http://www.i3s.unice.fr

A survey of RDB to RDF translation approaches and tools

Franck Michel [a], Johan Montagnat [a] and Catherine Faron-Zucker [a]
[a] *Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France*
E-mail: franck.michel@cnrs.fr, johan.montagnat@cnrs.fr, faron@i3s.unice.fr

Abstract. Relational databases scattered over the web are generally opaque to regular web crawling tools. To address this concern, many RDB-to-RDF approaches have been proposed over the last years. In this paper, we propose a detailed review of seventeen RDB-to-RDF initiatives, considering end-to-end projects that delivered operational tools. The different tools are classified along three major axes: mapping description language, mapping implementation and data retrieval method. We analyse the motivations, commonalities and differences between existing approaches. The expressiveness of existing mapping languages is not always sufficient to produce semantically rich data and make it usable, interoperable and linkable. We therefore briefly present various strategies investigated in the literature to produce additional knowledge. Finally, we show that R2RML, the W3C recommendation for describing RDB to RDF mappings, may not apply to all needs in the wide scope of RDB to RDF translation applications, leaving space for future extensions.

Keywords: Semantic Web, Database semantics, Database integration, RDB-to-RDF Mapping

1. Introduction

Making data hosted in relational databases (RDB) accessible to the semantic web has been an active field of research during the last decade. Converting relational data into RDF or exposing relational data so that it can be queried through the SPARQL[1] query language for RDF, is often referred to as the "RDB-to-RDF" process. In September 2012, the publication by the W3C of the R2RML recommendation [20], a standard language to describe mappings between a relational database and an equivalent RDF dataset, has marked a new step towards the actualization of the web of data. R2RML encourages RDB-to-RDF tool developers to comply with a standard mapping language. Data providers should benefit from the adoption of this common language, allowing them to decouple relational data integration problems from specific tools or approaches, and ensuring sustainability.

Many RDB-to-RDF techniques and corresponding tools have been proposed over the last years. In spite of the observed convergence of several of them towards R2RML, data providers willing to publish their data in a machine-readable format may find it difficult to make a choice. Firstly, different techniques convey different philosophical approaches (e.g. focus on ontology learning, mapping language design, query engine design...) which have implications on the way the relational data is exposed. Secondly, choosing the R2RML recommendation does not answer all the questions: like any other language, R2RML has some limitations with regards to the types of mappings that can be expressed. Besides, as an implementation-independent mapping language, R2RML does not address some common questions that occur when translating relational data into RDF, such as the implementation of the translation process, or the way the translated RDF data is accessed. Lastly, existing reviews of RDB-to-RDF approaches often provide very brief descriptions of the tools developed, making them difficult to compare. Operational questions such as the choice between the conversion of relational data into RDF repositories and the real-time use of the native relational databases, the choice of access and querying means to the exposed data, tools sustainability, etc., are hardly addressed.

[1] http://www.w3.org/TR/sparql11-overview/

1.1. Motivations for RDB-to-RDF Translation

In order to grasp the diversity of RDB-to-RDF approaches, it is useful to understand the motivations of the RDB-to-RDF studies and projects, which led to the emergence of R2RML. We identified three common needs often targeted: accessing data from the deep web, linking data, and integrating multiple heterogeneous data sources. We describe them in the following.

The "deep web", as opposed to the "surface web", is a part of the web content that is hardly indexed by standard search engines. It refers to the data hidden in unstructured documents (images, scans), semi-structured documents (CSV files, PDF files...), or structured data sources (relational databases, XML databases, NoSQL databases, LDAP directories...) that standard web tools cannot browse, but that are only accessible through query forms. As an illustration, in 2007, 70% of web sites were backed up by RDBs, which contained 500 times more data than directly available [34]. Making this huge amount of data available in a machine-readable format is expected to create opportunities for novel applications and services. In this regard, RDF is a powerful pivot format. Yet, in order to ensure the sustainability of the applications that were developed along with the data they exploit, and to leverage the properties engineered into RDB systems for decades (scalability, ACID[2] properties, security and performance optimizations), the data should remain hosted and delivered by the legacy RDBs, hence the need for RDB-to-RDF techniques that can access relational data and convert it into RDF triples.

Linking open data to other related pieces of data increases its value. From this simple statement, the Linked Data principles, proposed by Tim Berners-Lee [9], recommend best practices for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF. Driven by these recommendations, the Linking Open Data[3] community project aims at extending today's web by publishing various open data sets in the RDF model, and setting RDF links between data sources. In other words, it intends to solve the semantic web chicken-and-egg dilemma, stating that a critical mass of machine-readable data must be available for novel mash-up applications to arise. Such applications should create added-value by repurposing data sets, using the data in some new way, possibly beyond what data providers may have initially expected. In this regard, the success of the Linking Open Data project largely depends on the accessibility of the deep web data, and the availability of RDB-to-RDF tools to help publish the existing relational data into RDF.

Integrating heterogeneous data has become a major challenge in several domains [46,14]. For instance, in neurosciences it is increasingly needed to connect, make sense of, and search across heterogeneous data and knowledge describing different organization scales (molecules, proteins, genes, cells, physiology, behaviours...) [1]. The first major step to integrating heterogeneous relational data sources is to make their semantics explicit. Relational schemas usually convey no or poor semantics. To some limited extent, implicit semantics can be figured out from integrity constraints or usual database design patterns such as n-ary relations and inheritance. But additional semantics is frequently encoded in the application exploiting a relational database, for instance by means of domain specific rules. Moreover, relational schemas are often fine-tuned and customized for performance reasons. This results in mixing data semantics with technical concerns, making it even more difficult to figure out the original data semantics. As a result, in order to tackle the challenges of data integration in translational science, data integration techniques have to capture and expose its semantics in an explicit and machine-readable manner. Using RDF as a format for representing relational data appears as a powerful and promising method to achieve such data integration, in which RDB-to-RDF methods will play a key role.

1.2. Previous Works

Facing the large variety of RDB-to-RDF mapping initiatives, several studies have been conducted to compare approaches and techniques.

In 2007 the W3C created the RDB2RDF Working Group[4] to standardize languages for mapping relational database schemas into RDF and OWL. Sahoo et al. [47] conducted a wide scope review, addressing theoretical articles, proofs of concept, domain-specific projects as well as generic mapping tools. The goal of this survey was not to get into the details of each approach, but to provide the RDB2RDF Working Group with a comprehensive overview of the different approaches that had been investigated so far, in order to serve as a basis for the definition of R2RML.

[2] atomicity, consistency, isolation, durability

[3] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[4] http://www.w3.org/2001/sw/rdb2rdf/

Hert et al. [36] proposed a feature-based comparison framework that they have applied to state of the art mapping languages. It is derived from the use cases and requirements described by the W3C RDB2RDF Working Group [5]. The mapping languages are sorted into four categories: direct mapping, read-only general-purpose mapping, read-write general-purpose mapping, special-purpose mapping. This paper focuses on the comparison of the mapping language features and expressiveness, and it does not address the implementations proposed by their authors or the way queries are rewritten.

In 2010, Spanos et al. [53] wrote a comprehensive review of RDB-to-RDF methods, with a particular effort to classify them among disjoint categories: creation of database schema ontology, creation of a domain-specific ontology using either database model reverse engineering or database expert knowledge, and definition or discovery of mappings between a relational database and an existing ontology. Then, features are explored within each category, such as data accessibility or ontology language used. Nevertheless, the paper does not detail the specific features and expressiveness of the mapping languages, and the authors put the stress specifically on the creation and alignment of ontologies.

Sequeda et al. [50] surveyed methods that apply Direct Mapping principles to automatically translate a relational database into RDF. They studied existing methods to extract ontological knowledge as RDFS or OWL data from SQL DDL (Data Description Language) representations. This ranges from simple approaches (table to class, column to property) to more advanced ones that try to discover relations such as many-to-many, subsumption, meronymy, symmetric and transitive relations, and SQL features such as integrity constraints, checks and triggers. Finally, the authors extend the work of Tirmizi et al. [55], that exploits all the possible combinations of primary and foreign keys in relational tables. One of their conclusions is that the quality of an ontology resulting from a direct mapping highly depends on the richness of the SQL schema with respect to its encoding of domain semantics.

Also to be mentioned, Sequeda proposed a short high-level review[5] of major RDB-to-RDF mapping products, along with a brief description of each of them. Unlike the two previous reviews that almost exclusively focus on academic works, this blog article reviews products either from the academic world or from the industry. Hence, although very succinct, this work broadens the scope towards the industrial approaches.

To our knowledge, no RDB-to-RDF study has been conducted since R2RML was published in 2012. As a result, none of the aforementioned articles reviewed R2RML compliant tools.

1.3. Goal of this work

In this paper, we propose a detailed review of state of the art RDB-to-RDF tools, either academic or industrial. Especially, we take into account R2RML implementations that recently emerged. We make a specific focus on the specificities of each technique, and we describe, as much as possible, the capabilities of the different tools studied. Taking into account practical concerns, we specifically consider end-to-end projects that delivered operational tools to implement the RDB-to-RDF process, avoiding proofs of concepts and early prototypes. A classification is proposed to identify the different approaches along three major axes that span from mapping design to service deployment concerns: (i) mapping description (type of mapping, expressiveness); (ii) mapping implementation (how and when the data is converted into RDF), and (iii) data retrieval method (query-based, linked-data). These categories are well identified in the literature. They avoid overlaps, they can be identified non-ambiguously and they cover the major concerns motivating RDB-to-RDF transformation. We also provide information on the project maturity and sustainability when available.

2. Classification of RDB-To-RDF Approaches

2.1. Mapping Description

The mapping description refers to the way the mapping between a relational database and an RDF representation of it is described. It is generally driven by the fact that the resulting mapping should either come up with an ad-hoc ontology that reflects the relational schema (*Direct Mapping*), or comply with existing well-defined domain-specific semantics, by reusing domain ontologies and possibly entailing more complex mappings (*Domain Semantics-Driven Mappings*).

In the literature, *Direct Mapping* is frequently used as a synonym of *Automatic Mapping*, and *Manual Mapping* as a synonym of *Domain Semantics-Driven*

[5] http://semanticweb.com/relational-database-and-the-semantic-web_b16083

Mapping, although this happens to be misleading: a direct mapping is generally created automatically and later customized manually. Yet, despite the manual edition, it can remain nothing more than a direct mapping. Conversely, the domain semantics-driven mapping is often called manual mapping, although an automatically generated direct mapping is frequently used as a starting point to more complex domain-specific mappings. In addition, some approaches attempt to augment direct mapping by automatically discovering domain semantics, we shall refer to this as the augmented direct mapping. Consequently, hereafter we shall sort the approaches along three distinct categories: *Direct Mapping*, *Augmented Direct Mapping* and *Domain Semantics-Driven Mapping*. The terms *manual* and *automatic* will be used for exactly what they mean: the manual edition of a mapping by a human agent, vs. the automatic generation of a mapping by a program.

Direct Mapping

The direct mapping approach intends to convert relational data into RDF in a straightforward manner, by making explicit the semantics encoded in the relational schema. It involves the automatic creation of URIs following simple rules, such as those defined by Tim Berners-Lee [8]:

- table-to-class: a table is translated into an ontological class identified by a URI whose construction follows the pattern "namespace/database/table";
- column-to-property: each column of a table is translated into an ontological property whose URI follows the pattern "namespace/database/table/-column";
- row-to-resource: each row of a table is translated into a resource whose type is the class representing the table and whose URI is formed by using the table's primary key: "namespace/-database/table/primaryKey" or "namespace/database /table#primaryKey";
- cell-to-literal-value: each cell with a literal value is translated into the value of a data property;
- cell-to-resource-URI: each cell with a foreign key constraint is translated into a URI which is the value of an object property.

Applying this set of rules automatically creates an ad-hoc RDFS or OWL vocabulary reflecting the structure of the relational schema. The ontology thus created by the direct mapping process is referred to as the *database schema ontology* in [53], while ontology learning approaches often use the term *local ontology*. To avoid exposing unnecessary or sensitive data such as passwords, most approaches automatically generate a first mapping that can be manually customized to some extent. Some specific cases such as multi-column primary keys and tables with no primary key are also generally addressed.

The Direct Mapping method typically applies when no ontology suitably describes the domain of the relational database, or when the goal is to rapidly make data sources available in a web machine-readable format, with little concern for semantic interoperability. Direct Mapping can also address versatile environments in which databases may appear and disappear frequently with no time for manual alignment [22]. When semantic interoperability is required, ontology alignment methods can be used later on to align the local ontology with existing domain ontologies.

The W3C recommendation "A Direct Mapping of Relational Data to RDF" specifies direct mapping good practices [2]. It essentially proposes a formalization of the rules enounced by Tim Berners-Lee.

Augmented Direct Mapping

Augmented Direct Mapping targets to improve the quality of a direct mapping by the automatic detection of common database design patterns that can convey domain semantics. For instance, many-to-many relations are suggested by tables in which all non primary key columns are foreign keys to other tables; nullable/not nullable columns can be converted into OWL cardinality constraints; implicit subclass relationships are often suggested by a primary key used as a foreign key [18]. In the latter case though, the literature argues that, in the context of databases not in the third normal form, this pattern may reveal a vertical partitioning (splitting of a table into several smaller tables for performance concerns) rather than a subsumption relationship [55,50].

Some semi-automatic approaches propose an iterative process in which a domain expert validates or dismisses proposed mappings. Classes may be refined based on the detection of lexical clues in the column names or data redundancy suggesting categorization patterns [16]. Additionally, some research questions remain open with regards to the possibility of translating relational database triggers into additional knowledge in the form of semantic rules.

Domain Semantics-Driven Mapping

The direct mapping approach is hardly sufficient in real world applications, in which the data semantics lies outside the RDB schema, in domain-specific

rules encoded in the application that exploits the database. This is outlined in a short but enlightening feedback from the Ordnance Survey of Great Britain: "*databases are rarely good descriptions of a domain, being the result of both performance optimisation processes and contingent maintenance history. And, in any case, the schema itself will not support a full description of the domain, other relevant relationships often being buried in code or in the encoding of various attributes*" [26]. To overcome these limitations, the domain semantics-driven mapping approach applies when the relational database must be translated using classes and properties of existing ontologies. The database and the ontology may have been designed separately, and their similarity level may be low. A typical use case is the alignment of a legacy database with an existing ontology describing or referring to the same domain of interest.

Domain semantics-driven systems rely on mapping description languages to allow the description of expressive mappings, able to bridge the conceptual gap between RDB and RDF. Mapping description languages may support various features listed in Table 1. In any case, those languages generally implement two different strategies:

(i) The mapping description essentially relies on SQL queries to present the data as RDF triples. The expressiveness of mappings is therefore constrained by that of the SQL flavour used; in other words, complex cases that would require a richer expressiveness cannot be addressed, unless they are supported by extensions of a specific RDBMS. On the other hand, the rewriting of a query on the target RDF data (generally a SPARQL query [33]) into SQL is almost straightforward, and the query execution can benefit from the native database optimizer. Besides, the great popularity of SQL facilitates the adoption of the mapping language by data providers who do not need to learn a new mapping language.

(ii) The mapping description uses a specific dedicated language. In this approach, the query rewriting process is more complex as it does not rely on SQL queries written by the person who wrote the mapping. The mapping is not constrained by the expressiveness of SQL. As a result it can be extended in order to meet specific complex needs such as keyword search, regular expression matching, natural language processing, data mining, etc. Nevertheless, it must be underlined that most existing projects hardly reach the expressiveness of SQL (for instance, aggregation and grouping are not always possible although they are natively supported by SQL).

Some mapping languages such as R2RML and D2RQ use both strategies simultaneously: they are able to complement SQL snippets with specific mapping descriptors.

2.2. Mapping Implementation

Given a mapping description, that is, the set of rules that map a relational model to a target ontology, the mapping implementation refers to the way database tuples are translated into ontological instances (individuals). Two methods can be applied: *data materialisation*, or *on-demand mapping*.

Data Materialisation

Data materialisation is the static transformation of the source database into an RDF representation, like in warehouse approaches. Mapping rules are applied to the whole content of the database to create an equivalent RDF graph. For this reason, it is also referred to as "graph dump", "graph extraction" or "RDF dump". When the materialisation process completes, the resulting RDF graph can be loaded into a triple store and accessed through a SPARQL query engine. This whole process is often referred to as the *Extract-Transform-Load* (ETL) approach that conveys the idea of data materialisation and loading into a triple store.

With this approach, the number of data sources that can be integrated is only limited by the triple store and query engine capacity. Besides, a major advantage of the materialisation is to facilitate further processing, analysis or reasoning on the RDF data, including its linking to the Linked Open Data and the execution of heavy inference rules on it. Indeed, as the RDF data is made available at once, third party reasoning tools can be used to apply complex entailments. Later on, complex queries can be answered without compromising run-time performances since the reasoning has been performed at an earlier stage.

Several limitations are to be noticed though. This solution hardly supports very large data sets, as the size of the graph produced may exceed memory capacity. Another limitation concerns the way to deal with outdated data: in the context of an application that updates the relational data frequently, the materialized RDF graph may be rapidly outdated. A solution is to run the extraction process periodically, which raises the question of compromising between the cost of materializing and reloading the graph, and the tolerance of the application to outdated data.

On-Demand Mapping

Table 1
Features of mapping languages

Feature description	Feature description
generation of user defined unique Ids	Ability to generate URIs of resources beyond the simple use of primary key values: reusing and combining column values, allowing for conversion tables, etc.
logical table	Ability to read tuples not only from tables but also from SQL views or from the result of an SQL query.
column selection (also called projection)	Ability to select only a subset of the columns of a table to translate. This is a very basic feature, almost a minimum pre-requisite of any RDB-to-RDF tool.
column renaming	Ability to map a column to an RDF property with a different name. This is not always possible in a direct mapping but quite obvious in a domain semantics-driven mapping.
select conditions	Ability to translate only a subset of the tuples of a table using a select-where condition.
vocabulary reuse	Ability to map relational entities to instances of existing vocabularies and ontologies. This is the main difference between domain semantics-driven mapping and direct mapping approaches.
1 table to n classes	Ability to use the values of a column as a categorization pattern: tuples of the table will be translated into instances of different ontological classes based on the value of this attribute. This feature can be seen as an extension of the "select conditions" feature as it results in not only filtering out rows, but the filter helps selecting rows to be converted into instance of one class or another.
many-to-many relation to simple triples	Many-to-many relations are usually implemented in relational databases as a join table in which all columns are foreign keys to other tables (n-ary relations). This feature refers to the ability to translate many-to-many join tables into simple triples, as opposed to a basic direct mapping in which the join table will be translated into a distinct class.
blank nodes	Ability to generate blank nodes and refer to them within the graph produced during the translation process. Blank nodes can be used for instance to translate a table without a primary key.
data types	Ability to handle relational data types consistently with RDF data types per SQL-XSD mapping.
data transformation	Ability to apply transformation functions to the values before generating the RDF triples. This can be used to perform complex type conversion, compute a value using several columns, and applying methods such as string manipulation functions, decimals type conversions, etc.
named graphs	Ability to create not only one default RDF graph but also multiple named graphs within a single mapping definition.
user-defined namespaces	Ability to declare and use namespace prefixes.
static metadata	Ability to attach static metadata (such as licensing or provenance information) to the produced graphs, and possibly to all RDF entities or instances of a certain class.

Conversely to the data materialisation method, the on-demand mapping approach is the run time evaluation of queries against the relational data. In this model, the data remains located in the legacy database. Whatever the way the converted data is accessed, queries to the target RDF data must be rewritten into SQL at query evaluation time.

The advantages and drawbacks of this approach are the opposite of the materialisation approach. It is well suited in the context of very large data sets that would hardly support centralization due to resource limitations. It guarantees that the returned data is always up to date since no copy of the RDF data is made. Besides, it allows for the enforcement of access control policies implemented in the RDBMS.

On the other hand, query performance can be severely penalised if entailment regimes must be implemented [47], or if many data sources are to be integrated together. It has been suggested that in some cases, the expressiveness of SPARQL queries should be limited, in order to be processed by on-demand mapping systems: in particular, a variable in the predicate position `(resourceA ?r resourceB)` or a variable in the object position representing a class `(resourceA rdf:type ?c)` can lead to "union bomb" issues [28,27].

2.3. Data Retrieval

Independently of the way a mapping is implemented, the RDF data can be retrieved using two main query implementation methods: by sending a query

to a query processing engine or using the linked data paradigm. The choice of the method largely depends on how the data should be exploited, as detailed below.

Query-based access

RDF data is retrieved by means of a query, generally expressed in SPARQL, the standard query language for RDF. A SPARQL query engine may be accessed through an API[6], or exposed through the SPARQL protocol [29] resulting in a SPARQL endpoint.

In the data materialisation approach the SPARQL query engine evaluates a query against the RDF repository in which the materialised RDF data has been loaded. In the case of the on-demand mapping, it evaluates the query against a relational database. This involves rewriting the SPARQL query into SQL and, conversely, translating SQL results into equivalent SPARQL results according to the mapping.

Early approaches proposed query languages other than SPARQL. Nevertheless they were deprecated by the standardization of SPARQL. Additionally, some RDBS providers have proposed alternative solutions by integrating a SPARQL query evaluation engine within the native RDBS evaluation engine. This is the case of SPASQL[7] that allows for the execution of SPARQL queries within SQL statements.

Linked Data

During the RDB-to-RDF process, each logical relational entity translated into RDF is assigned a unique URI that identifies it in the data graph. According to the principles of the Linked Data [9], it should be possible to *dereference* any such URI by submitting an HTTP GET request with this URI, as if it was a URL. The HTTP response should provide a representation of the entity identified by the URI. The output format of the data description is generally agreed during a regular HTTP content type negotiation procedure between the client and the web server.

Two access methods are advised in [45], in relation to good practices for defining URIs:

(i) For an informational resource, a simple web lookup with HTTP content negotiation will return a representation of the entity (in XHTML, RDF/XML, N3, JSON, etc.).

(ii) If the URI refers to a non-informational resource, i.e. an abstract concept or a physical object, it should not be dereferenced directly (dereferencing the URI of a person cannot return this physical person). An HTTP GET with such a URI should return HTTP status *303 See other*, providing the URI of an informational resource that relates to or is a representation of the non-informational resource, e.g. an HTML document describing a person. This recommendation may somehow be relaxed for technical concerns such as the extra network traffic entailed by the additional HTTP request, or time consuming definition of multiple URIs.

In the RDB-to-RDF context, the term *Linked Data* sometimes refers to the ability to dereference not only a resource URI, but also a URI to a logical entity of the source database. For instance, dereferencing the URI of a database will return a human-readable list of classes corresponding to the tables, dereferencing the URI of a class will return URIs of instances of this class. Closer to the spirit of the Linked Data, D2RQ dereferences a class URI by providing a few triples stating that it is a database or a class, including *rdfs:seeAlso* statements when relevant. Such methods can be exploited by crawlers of external search engines to index the content of the database.

Graph Dump Access

One could argue that a third data retrieval method exists: graph dump over HTTP access. In this method, a client performs an HTTP GET request to the database URI and retrieves the entire RDF graph at once. The "SPARQL 1.1 Graph Store HTTP Protocol" [39] standardizes this method within the wider scope of graph management methods. Nevertheless, none of the tools that we have studied implements the graph dump over HTTP access method. This can be explained by the fact that the data materialisation process produces a graph serialization, generally in the form of an RDF file. Hence, using the graph dump over HTTP access method would be just another way of getting an alternative representation of the same graph from an HTTP endpoint. Moreover, in conjunction with the on-demand mapping implementation, the dump of large graphs may induce significant performance issues. Therefore, in the following, we shall not consider the *Graph Dump over HTTP* access in the list of data retrieval methods.

3. R2RML

R2RML [20] is a generic language to describe a set of mappings that translate data from a relational database into RDF. It is the result of preliminary works held by the W3C. Starting from an initial proposal in

[6] http://www.w3.org/wiki/SparqlImplementations
[7] http://www.w3.org/wiki/SPASQL

2007 [24], the W3C RDB2RDF Incubator Group[8] ran a comprehensive survey of existing approaches [47] and described high level characteristics that a language should cover to map relational databases to RDF [3]. Finally, the RDB2RDF Working Group standardized the R2RML mapping language in 2012. As a standard, R2RML has a particular importance as it steers the development of many RDB-to-RDF tools. Yet, it only covers the mapping description language, but does not recommend any implementation. Compliant tools may therefore adopt completely different strategies.

3.1. R2RML Features

The W3C RDB2RDF Working Group proposed a list of 11 mandatory or optional requirements for R2RML [5]. Table 1 provides a detailed description of each of these features.

Mandatory features state that R2RML must (i) support both the direct mapping and the domain semantics-driven mapping (called transformative mapping); (ii) provide sufficient information for a processor to support both the on-demand mapping (rewrite SPARQL queries into SQL) and the data materialisation. Other mandatory features are: generation of unique identifiers, support of data types conversions, column renaming, many-to-many relation to simple triples, 1 table to n classes.

Optional features include data transformation, named graphs, namespace declaration, static metadata. Additionally, the "update logs" feature is also listed as a "nice-to-have feature": the mapping should provide extension points to support the creation of update logs of relational data. The final R2RML recommendation includes all features listed above with the exception of the update logs and the data transformation that is left to the capabilities of SQL in terms of string or number manipulations (no complex data transformation method is specified).

Table 1 includes three additional features that were not explicitly addressed in [5], but that are part of R2RML: select conditions, column selection and blank nodes. The "select conditions" feature is implicit in R2RML as it is a pre-requisite of the "1 table to n classes" feature. Similarly, the "column selection" feature is implied by the more global support of the transformative mapping.

[8] http://www.w3.org/2005/Incubator/rdb2rdf/

3.2. R2RML Mapping Description

An R2RML mapping document is called an R2RML *mapping graph*. It is written in RDF with the Turtle syntax[9]. In the following, we describe R2RML main components along with a running example. Let a database describe courses and participating students. Table COURSE has two columns: ID (primary key) and TITLE. Table PARTICIPATES has two columns: SID (a student identifier), and CID (foreign key to column ID in table COURSE). The R2RML mapping in Listing 1 maps this database to RDF triples using an existing ontology with classes `co:Course`, `co:Student`, a `co:participatesIn` object property and a `co:title` data property .

A mapping consists of several *TriplesMaps*, each one specifying how to map rows of a logical table to RDF triples. The logical table may be a table, an SQL view, or the result of any valid SQL query. Our example defines two TriplesMaps, `Course` with logical table COURSE and `Participates` with logical table PARTICIPATES.

A TriplesMap is composed of exactly one *SubjectMap* and any number of *PredicateObjectMaps*. For each row of the logical table, the SubjectMap generates the subject URI, for instance using the logical table primary key. The SubjectMap in TriplesMap `Course` defines a subject URI as the concatenation of namespace `http://univ.org/course/` with the value of column ID. PredicateObjectMaps consist of *PredicateMaps* and *ObjectMaps*. Triples are produced by combining the subject map with each predicate from the PredicateMap and each value from its associated ObjectMap. In TriplesMap `Course`, each subject URI will have a predicate `co:title` with literal value read from column TITLE.

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix co: <http://courses.org#>.
@base <http://example.com/base#>.

<Course>
 rr:logicalTable [ rr:tableName "COURSE" ];
 rr:subjectMap [
  rr:template "http://univ.edu/course/{ID}";
  rr:class co:Course;
 ];
 rr:predicateObjectMap [
  rr:predicate co:title;
  rr:objectMap [ rr:column "TITLE" ];
 ].
```

[9] http://www.w3.org/TR/turtle/

```
<Participates>
 rr:logicalTable
  [ rr:tableName "PARTICIPATES" ];
 rr:subjectMap [
  rr:template
   "http://univ.edu/student/{SID}";
  rr:class co:Student;
 ];
 rr:predicateObjectMap [
  rr:predicate co:participatesIn;
  rr:objectMap [
    rr:parentTriplesMap <Course>;
    rr:joinCondition
     [ rr:child "CID"; rr:parent "ID". ];
  ];
 ].
```

Listing 1: Example of an R2RML mapping

In TriplesMap `Participates`, object property `rr:subjectMap` defines URIs for students, while `rr:predicateObjectMap` describes an SQL inner join between tables COURSE and PARTICIPATES. This is denoted by the value of property `rr:objectMap` having two specific properties: property `rr:parentTriplesMap` refers to TriplesMap `Course` as the parent in the join condition, and property `rr:joinCondition` defines the columns to join: PARTICIPATES.CID and COURSE.ID.

Assuming table COURSE contains one course with identifier 1 and title "Semantic Web" and table PARTICIPATE contains one student with identifier 2 who participates in course 1, the above R2RML mapping will produce four RDF triples:

```
<http://univ.edu/course/1>
     a co:Course;
     co:title "Semantic Web".
<http://univ.edu/student/2>
     a co: Student;
     co:participatesIn
     <http://univ.edu/course/1>.
```

By default, all RDF triples are in the default graph of the output dataset. A TriplesMap can contain *GraphMaps* that place some or all of the generated triples into named graphs.

The R2RML recommendation specifies that an R2RML processor may include an R2RML default mapping generator complying with the W3C direct mapping rules [2].

3.3. Implementations

The first R2RML recommendation was issued Sept. 27th 2012. Several candidate implementations have been evaluated against compliance tests described in the *R2RML and Direct Mapping Test Cases*[10], comprising 24 Direct Mapping tests and 62 R2RML tests. Results are reported in the *RDB2RDF Implementation Report*[11].

4. RDB-to-RDF tools

This section describes RDB-to-RDF tools classified along the axis defined in section 2. Section 4.1 presents the tools that comply with R2RML, while section 4.2 describes the tools that propose their own mapping language. In each section, tools are listed by alphabetical order. Table 2 (section 4.3) summarizes the information provided here after.

4.1. R2RML-Compliant Tools

Below we consider the R2RML candidate implementations and their results with regards to the R2RML test cases. Oracle Spatial and Graph 12c, released in July 2013, supports R2RML. Yet, at the time of writing, it has not been officially tested against the R2RML test cases and thus is not yet mentioned in the RDB2RDF Implementation Report.

4.1.1. DB2Triples

DB2Triples[12][13] is an implementation of R2RML and the W3C Direct Mapping based on the Working Draft of May 29th 2012[14]. It is developed by the Antidot[15] company as part of a larger software suite. DB2Triples is delivered as a Java library, available under the LGPL 2.1 open source licence, and validated with MySQL and PostgreSQL back-ends. It supports the data materialisation mapping implementation, but provides no data retrieval method. The materialized graph can be serialized in RDF/XML, N3, N-Triples or Turtle. In the direct mapping mode, DB2Triples can

[10] http://www.w3.org/TR/2012/NOTE-rdb2rdf-test-cases-20120814/
[11] http://www.w3.org/TR/2012/NOTE-rdb2rdf-implementations-20120814/
[12] http://www.antidot.net/fr/Actualites/Produit/Antidot-fournit-db2triples-en-Open-Source
[13] https://github.com/antidot/db2triples
[14] http://www.w3.org/TR/2012/WD-rdb-direct-mapping-20120529
[15] http://www.antidot.net/

optionally apply queries from a SPARQL file in order to transform the graph. This may be used to produce additional triples using deduction rules in the form of SPARQL INSERT clauses.

Compliance: DB2Triples failed on one of the 62 R2RML test cases regarding the detection of a non-conforming mapping.

Sustainability: DB2triples version 0.9.9 was released in 2012. No indication as to its support is provided, one can expect Antidot to maintain it as part of Antidot's software suite.

4.1.2. Morph-RDB

Morph-RDB[16] is an implementation of R2RML, developed by the developers of R2O and ODEMapster. It supports the data materialisation (called data upgrade mode) and the on-demand mapping using SPARQL queries. In [43], authors describe the method they developed to enable R2RML-based SPARQL to SQL query translation.

Morph-RDB is developed in Scala and Java and is available under the Apache 2.0 open source licence. The development is ongoing. Authors are also working on two other versions of Morph, that rely on R2RML although they address non-relational databases: Morph-streams deals with streamed data typically produced by sensors, and Morph-GFT queries Google Fusion Tables[17] like relational tables.

Compliance: Morph-RDB failed on 8 of the 62 R2RML test cases regarding the generation of a default mapping for tables without a primary key (2 tests), data type conversions (3 tests), different blank nodes generated for different rows when they should be the same, detection of non-conforming mappings (2 tests). R2RML named graphs are not supported in the on-demand mapping but is supported in the data materialisation.

Sustainability: developers intend to continue the support and evolution of Morph-RDB, possibly extending it beyond the official R2RML specification, e.g. by adding support for Google Fusion Tables. Little documentation is available, but extensive usage examples are provided.

4.1.3. Oracle Database 12c

Oracle Spatial and Graph[18] (formerly Oracle Semantic Technologies) [41,40] is an option of Oracle Database Enterprise Edition (EE). Version 12c, released in July 2013 comes with the *RDF Semantic Graph* data management and analysis features, that support RDB-to-RDF conversion.

RDF Semantic Graph mainly focuses on the storage, simultaneous querying and reasoning on relational and RDF data. The RDF graph store can scale up to billions of triples, supports graph versioning and the semantic indexing of documents. SPARQL graph patterns can be included within an SQL query in order to join RDF and relational data. Extensive RDFS/OWL2 capabilities support the simultaneous reasoning on RDF and relational data. The security model can enforce restrictions at different levels, from graph to triple granularity.

RDB-to-RDF conversion is supported since version 12c by providing RDF views on relational tables, SQL views, and SQL query results. R2RML and the W3C Direct Mapping are supported. The RDF view can be queried through SPARQL 1.1.

Oracle Spatial and Graph exploits several enterprise features such as table compression (optimize disk space and memory usage), partitioning (performance, scalability), and *Real Applications Clusters* (availability, scalability). As a result, using Oracle Spatial and Graph requires the acquisition of licenses for Oracle Database EE and Partitioning option.

Compliance: no result is available as to the R2RML compliance tests, consequently, Table 2 shows question marks for all the features.

Sustainability: RDB-to-RDF is a newly added feature in release 12c. No information is available on its support by Oracle in the future.

4.1.4. RDF-RDB2RDF

Perl RDF is an open source software library delivered under the GPL license. It is a very complete library including the following features: RDF store in memory, support for relational databases (MySQL, PostgreSQL) and SQLite, SPARQL 1.1 query processor and endpoint, Linked Data server, RDFa parser, WebID (FOAF+SSL) + ACLs, GRDDL, Microformats, HTML5.

RDF-RDB2RDF[19] is an additional library, developed on top of Perl RDF, that implements R2RML and the W3C Direct Mapping based on the Working Draft of May 29th 2012. RDF-RDB2RDF implements the data materialisation method. The materialized RDF data can be loaded into the SPARQL endpoint provided by Perl RDF, thus falling in the ETL approach.

[16] https://github.com/fpriyatna/odemapster/wiki
[17] http://www.google.com/drive/apps.html#fusiontables
[18] http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/rdfsemantic-graph-1902016.html
[19] https://metacpan.org/release/RDF-RDB2RDF

Compliance: RDF-RDB2RDF failed on 12 of the 62 R2RML test cases (either with PostgreSQL or SQLite) regarding data type conversions (5 tests), management of joins and inverse expression (4 tests), named graphs, logical table named column, special chars and backslashes.

Sustainability: the implementation of R2RML and the direct mapping dates back to version 0.006 delivered in June 2012. However Perl RDF is actively maintained, version 0.008 was delivered in September 2013.

4.1.5. Ultrawrap

Initially developed by the University of Texas in Austin, Ultrawrap[20] is now a commercial product of the Capsenta company, founded in 2011 as a spin-off of the University of Texas. It is based on SQL views to present relational data as RDF triples.

A local ontology is defined through the direct mapping. The RDF representation is implemented as a three-column SQL view (subject, predicate, object), defined as the union of all the queries that materialize all the RDF triples as defined by the local ontology [52]. Consequently, a SPARQL query can be simply rewritten into an SQL query on the SQL view. In [51], authors explain the need for two additional columns in the SQL views: the primary keys of the subject and the object. This change allows the native query optimizer to take advantage of existing indexes. Along with two other optimizations (the self-join elimination and the detection of unsatisfiable conditions), authors show that the query evaluation time is comparable to that of SQL queries written directly for the relational representation. Ultrawrap also supports the Linked Data retrieval method (HTTP GET requests).

Recently, support for R2RML and D2RQ mapping languages has been added. There is no description, however, of the way the mapping description is derived into the SQL view. We make the hypothesis that an R2RML/D2RQ document is compiled into an SQL view that reflects each triple map. The support of R2RML named graphs does not seem easy using only the SQL triple view. A GUI that is part of the tool suite helps align the local ontology with a domain ontology.

Compliance: Ultrawrap passed all R2RML test cases.

Sustainability: Ultrawrap was released to first beta customers in May 2012. No information is given as to future releases. Prices are available on demand only.

[20] http://capsenta.com/ultrawrap

4.1.6. Virtuoso Universal Server & Virtuoso's RDF Views

Virtuoso Universal Server[21] is a commercial and open-source comprehensive tool suite developed by OpenLink, designed to meet enterprise data management, access and integration needs. It comes with production-class features such as a relational database, clustering, data replication, RDF triple store, reasoning capabilities (entailment regimes), multiple data sources integration (SQL, RDF, XML, free text, CMS, aggregation feeds...). The Virtuoso Open-Source Edition[22] is a sub-set of the Universal Server. Limitations concern production-class features such as the clustering and data replication functions.

The RDF and SPARQL tool suite open source edition provides features such as: Object-Relational Database for SQL, XML, RDF, and Free Text; RDF store and SPARQL end-point; Web Application Server (see the full feature list[23]). In particular the *RDF Views of SQL data* functionality implements the RDB-to-RDF functionality. The open source edition only supports the Virtuoso-based relational database whereas the commercial edition supports most well known relational database systems.

The declarative *Meta Schema Language*[24] (MSL) is an extension of the SPARQL query language, meshed with Virtuoso's SPASQL functionality (SPARQL-inside-SQL). The mapping does not only involve an MSL document, but it is itself a repository in which mapping patterns can be grouped in named sets, and managed using operations such as create, drop, alter, etc. A rich web interface provides a wizard to automatically generate a direct mapping. R2RML support is achieved by the inclusion of a simple adaptor which basically translates R2RML syntax to Virtuoso's own Linked Data Views syntax, which can then be executed to create the Linked Data Views themselves.

The on-demand mapping is provided through Virtuoso SPARQL 1.1 endpoint. The data materialisation may be possible but this is not the goal of Virtuoso. The Linked Data retrieval method is also available and exposes any Virtuoso-housed data (SQL and XML data sources) as dereferenceable URI. RDF data may be retrieved as RDF/XML, JSON or N3 syntaxes.

Compliance: Virtuoso RDF Views did not pass 29 of the 62 R2RML test cases; those are in status "can-

[21] http://www.w3.org/wiki/VirtuosoUniversalServer
[22] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSIndex
[23] http://virtuoso.openlinksw.com/features-comparison-matrix/
[24] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSSQL2RDF

notTell", meaning that the test could not be run. No explanation is provided.

Sustainability: Virtuoso is actively maintained by the company OpenLink. Note that the DBpedia project is operated using a Virtuoso triple store.

4.1.7. XSPARQL

RDF/XML syntax provides a lot of variability in the serialization of an RDF graph. As a result, XSLT+XPath processors are inappropriate to integrate XML and RDF/XML data. The XSPARQL[25] query language was designed to address this difficulty, by combining XQuery and SPARQL for bidirectional transformations between RDF and XML: XSPARQL merges SPARQL components into XQuery FLWOR expressions. XSPARQL is typically designed to extract RDF data out of existing Web pages, allow an RDF-based client software to communicate with XML-based Web services, or even enrich an RDF graph with deduction rules described as RDF-to-RDF mappings.

Recently, Lopes et al. [38] have defined an RDB-to-RDF extension to XSPARQL, that provides the ability to embed SQL snippets (select-from-where) in XQuery FLWOR expressions. Thus, the relational data can be queried and transformed into XML or RDF. On top of this, an R2RML mapping document can be interpreted to produce appropriate XSPARQL construct expressions that materialize the RDF data. The originality of this approach is that it did not require the development of a specific module in the XSPARQL processing engine. Instead, the whole process is described in XSPARQL: the R2RML mapping document is read as input RDF data, XQuery FLWOR expressions parse it, query the database accordingly, and ultimately generate the RDF data using XSPARQL construct expressions.

XSPARQL comes with a prototype implementation developed in Java and distributed under the Apache 2 open source license. It implements the data materialisation approach. Supporting the on-demand mapping should be possible but is not straightforward: it would require designing a query rewriter to convert a SPARQL query into an XSPARQL query that would in turn make the appropriate SQL queries.

Compliance: XSPARQL passed all R2RML test cases, although discussions with the developers revealed that named graphs are not supported, and logical tables cannot refer to SQL queries (only relational tables and views are supported).

[25] http://xsparql.deri.org/

Sustainability: Version 0.5 was released in November 2012, supporting both Direct Mapping and R2RML. Several fixes were provided in early 2014.

4.2. Non-R2RML Tools

4.2.1. Asio Semantic Bridge for Relational Databases and Automapper

Asio Tool Suite[26] is a commercial product developed by BBN Technologies. It provides several components: Asio Semantic Query Decomposition (SQD), Asio Semantic Bridge for Relational Databases[27] (SBRD), Asio Semantic Bridge for Web Services, and two open source tools: Parliament and Snoggle.

Asio SBRD implements the RDB-to-RDF process. The BBN's Automapper tool applies D2RQ-based direct mapping rules to create the local ontology that describes the relational schema. The direct mapping is then augmented with OWL property restrictions to model data types and nullable/not nullable properties. Additional mapping rules between the domain ontology and the local ontology are described as a set of SWRL[28] rules. Asio SBRD implements the on-demand query mode: it rewrites SPARQL queries into SQL queries and applies query planning and optimization techniques. The data materialisation mode is not available. The precise set of supported mapping features is not available, consequently we did not include it in Table 2.

The Asio SQD module addresses the federation of multiple data sources: it breaks down SPARQL queries expressed using a domain ontology into sub-queries using the local ontology of each data source, and distributes these optimized sub-queries to the applicable data sources: relational databases, web services or triple stores.

4.2.2. D2R Server and the D2RQ language

The D2R Server[29] [12,13] is an open source academic project. It provides an integrated environment with multiple options to access relational data using different methods such as the SPARQL endpoint, Linked Data (content negotiation, HTTP 303 dereferencing), RDF dump. D2RQ supports both direct and domain semantics-driven mappings. The DR2Q declarative mapping language [19] is formally defined by an RDFS schema. It is the successor to the XML-

[26] http://bbn.com/technology/knowledge/asio_tool_suite
[27] http://bbn.com/technology/knowledge/asio_sbrd
[28] http://www.w3.org/Submission/SWRL/
[29] http://d2rq.org/

based D2R MAP language [11]. The mappings are expressed in RDF, but also largely rely on SQL fragments to express SELECT statements or to use aggregate functions. The D2RQ direct mapping automatically creates a local ontology. This direct mapping can be customized manually. Optionally, the direct mapping generated can comply with the rules proposed in the W3C Direct Mapping specification [2].

The Linked Data retrieval method is supported with HTTP content negotiation (RDF or XHTML), including HTTP status *303 See other*.

The automatic addition of `rdfs:seeAlso` properties and HTML hyperlinks leads to navigation pages containing lists of other resources of the same class, and to an overview page that lists all of these navigation pages. This overview page provides an entry point for external Web search engines to index the content of the database.

The performance varies depending on the access method and is reported to perform reasonably well for basic triple patterns, but there are limitations for graph patterns with filters and solution sequence modifiers.

Sustainability: D2R is a very active project. The last version released in June 2012 supports the W3C's Direct Mapping specification. The first release of DBpedia in 2007 was done using D2R Server, which since then migrated to Virtuoso RDF Views.

4.2.3. Datalift

Datalift[30] is an experimental research project aimed at helping users to publish and interlink their data sets on the web as Linked Data. It provides an integrated set of tools that ease the publication process of raw structured data coming from various formats (relational databases, CSV, XML, ...). The tools cover the following functions: selecting ontologies for publishing data, converting it to RDF by using the selected ontology, publishing the produced RDF data, interlinking it with other RDF datasets.

Datalift implements the ETL approach: the data is first materialised, loaded into the triple store, then it is retrieved through a SPARQL query engine or Linked Data navigation (URIs dereferencing based on content negotiation: RDF, CSV or XHTML).

The mapping description is done interactively through a web-based GUI: a user selects a data source that is automatically translated following the direct mapping method. Then, a set of modules helps align the produced RDF data on chosen ontologies: RDF-to-RDF transformation (by means of SPARQL queries), renaming of URIs (using regular expressions), conversion of strings into URIs, automatic discovery of links with other RDF datasets with the SILK[31] tool. As a result, although it starts with a Direct Mapping, the different modules provide enough flexibility to perform an a posteriori alignment with capabilities comparable to domain semantics-driven mapping approaches.

Sustainability: The project completed in March 2014. The Datalift association has recently been founded, which should guarantee a maintenance task force.

4.2.4. DB2OWL

DB2OWL is a proof of concept that was proposed by Cullot et al. [18]. In the domain of ontology learning, its goal is to automatically generate a direct mapping describing a relational database, then refine this mapping by exploiting relational schema characteristics: detect many-to-many join tables (translated into RDF triples) and concept subsumptions (when a primary key is also a foreign key in another table). The ontology produced is expressed in OWL-DL, while the mapping description is stored in an R2O document (see section 4.2.6).

DB2OWL is exclusively focused on the automatic creation of ontologies (classes and properties) that reflect the relational schema; it does not tackle the conversion of the relational data. It must be noticed that most direct mapping implementations are now able to handle many-to-many relation (although it is not part of the W3C Direct Mapping), whereas this is hardly the case of subsumption relations.

In [30], the authors of DB2OWL apply the on-demand mapping implementation to perform the integration of several relational databases using two different levels of ontologies: DB2OWL creates a local ontology for each database and a wrapper converts queries to a local ontology into SQL queries. At the top, user queries expressed on a domain ontology are mapped to queries on local ontologies.

Sustainability: DB2OWL is a prototype, with no more development since 2007. The source code is not available.

4.2.5. METAmorphoses

The METAmorphoses processor[32] transforms relational data into RDF using existing target ontologies. The mappings are specified in a declarative XML-based mapping language [54], and the RDF graph is produced at once (data materialisation) in an RD-

[30] http://datalift.org/en

[31] http://silk.semwebcentral.org/

[32] http://www.svihla.net/metamorphoses/METAmorphoses_processor/#documentation

F/XML output document. METAmorphoses is provided as a Java library under the LGPL open source license. Although quite simple, METAmorphoses is an effective tool. As other standalone applications, it does not require any complex deployment. Mappings can be implemented and tested easily and quickly.

The mapping language is organised as a two-layer data transformation model. The mapping layer describes how to map database tuples, selected by embedded SQL snippets, to RDF individuals using existing ontologies. The template layer describes how to serialize the mapped entities into RDF/XML.

METAmorphoses does not provide any data retrieval method, but relies on the ETL approach to query the materialized graph.

Sustainability: No update was provided since 2007.

4.2.6. R2O and ODEMapster

R2O is a declarative XML-based language [7] that allows the description of complex domain semantics-driven mappings between existing ontologies and relational elements (relations and attributes). It was initially designed to overcome weaknesses of D2R MAP [11], the predecessor of the D2RQ mapping language, for cases where the similarity between the relational database and the ontology is low. It addresses situations such as: 1-table-to-n-classes or joined-tables-to-1-class, that are commonly addressed by other projects such as D2RQ or Virtuoso. R2O also provides data transformation primitives such as string manipulations, arithmetic calculations, definition of order relations, expression of restriction on range of values, etc. To our knowledge, other RDB-to-RDF tools do not address such transformations, but one can argue that major RDB systems do cover such data manipulations through specific SQL extensions.

R2O is an evolution of eD2R [6], which addressed the mapping of lightly structured databases or databases not in first normal form. eD2R proposed complex transformations on field values based on techniques such as keyword search, regular expression matching, natural language processing and others.

The ODEMapster[33] query engine uses an R2O mapping document to either execute the transformation in response to a query expressed in the ODEMQL query language, or to apply the data materialisation approach (called massive upgrade). SPARQL and the Linked Data retrieval methods are not supported, but the ETL approach can be applied by loading the materialised RDF data into a third party triple store.

Sustainability: ODEMapster has been integrated as a plug-in into the NeOn[34] toolkit, an open source ontology engineering environment (ODEMapster plug-in still maintained in last toolkit version Dec. 2011). The EU ADMIRE project reused the NeOn toolkit and provided a distributed access to ODEMapster by using OGSA-DAI [42]. It is likely that the R2O language will no longer be maintained as authors have rewritten ODEMapster into the Morph-RDB R2RML-compliant product (see section 4.1.2).

4.2.7. RDBToOnto

The RDBToOnto[35] tool was designed in the context of the TAO project[36] (ended in early 2009) whose goal was to allow a fast and effective transition of existing legacy applications to ontologies.

RDBToOnto consists of a GUI-based extensible tool and a framework to ease the development and experimentation of "transitioning methods", i.e. ontology learning from relational databases. RDBToOnto proposes a semi-automated method: first, a direct mapping automatically creates a local ontology; then, the classes of this ontology are refined based on the relational data itself [16]: find lexical clues in the column names (matched against a predefined list of keywords e.g. "Type") or use data redundancy to discover categorization patterns. Another step allows for database optimization (removal of redundancies using the third party tool LATINO). The whole process is interactive (user defined rules) and iterative. The process completes with the production of the populated ontology: RDBToOnto implements the data materialisation approach.

RDBToOnto can be extended by means of connectors and converters to implement new learning methods. It does not specify any mapping language: constraint rules are stored in application specific project files and edited through the GUI only.

Sustainability: RDBToOnto has been used in a large real-world case study in aircraft maintenance[37], including mixing with existing ontologies, maintenance documentation annotation and WSDL annotation. No indication of use and maintenance outside of this context is found.

[33] http://neon-toolkit.org/wiki/ODEMapster

[34] http://www.neon-project.org/nw/Welcome_to_the_NeOn_Project

[35] http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto.html

[36] http://www.tao-project.eu/index.html

[37] http://videolectures.net/eswc08_cebrah_tla/

4.2.8. Relational.OWL

Peer-to-peer databases are volatile distributed databases, with frequently changing data and schema. Sharing data between such databases requires an exchange format that can be understood instantly, without requiring content negotiation or ontology alignment. The goal of Relational.OWL[38] is to enable such data sharing by providing an application independent representation technique expressed as an OWL ontology. Relational.OWL helps represent relational schema components by means of a local ontology (e.g. complying with direct mapping rules), as well as the relational data itself. It defines classes for tables, columns, primary and foreign keys, data types, and relations (properties) between those classes. OWL-full is required in order to allow the creation of an ontology with classes defined as instances of the Relational.OWL ontology classes. The use of OWL-full may be considered as a hurdle to the adoption of Relational.OWL, as it is non decidable. Nevertheless authors are "confident of most OWL reasoning tools being able to handle data and schema representations created using Relational.OWL" [21].

The Relational.OWL application is a GUI that automates the process of converting a database into its equivalent OWL full expression. No data retrieval method is provided, but based on Relational.OWL, De Laborda and Conrad [23] proposed a method to perform domain-semantics mapping: in a nutshell, the system first implements the ETL approach, then aligns the produced RDF data with the target ontology by processing SPARQL CONSTRUCT queries.

The advantage of expressing the schema of a relational database by building an ontology based on the Relational.OWL ontology is interoperability. The fact that two database schemas are described using the same base ontology helps compare classes with classes and properties with properties. However it does not solve alignment issues: for instance, it cannot help figure out that two columns from two databases bear the same semantics. Another advantage of Relational.OWL pointed out by De Laborda and Conrad [22] is that it keeps track of the relationship between the produced RDF data and the original database. Applications of this property could be the tracing of data provenance, the explanation of query results, or the assessment of confidence in query results.

Sustainability: Relational.OWL was not updated since 2006.

[38] http://sourceforge.net/projects/relational-owl/

4.2.9. SPASQL

SPASQL[39] is an open-source modified MySQL server, able to parse both SQL and SPARQL queries. The goal of providing MySQL with native support for SPARQL is to allow the same performance as for well-tailored SQL queries, by avoiding a complex rewriting phase [44]: SPARQL and SQL queries are compiled into the same data structures, and are then equally processed by the MySQL query processing engine. SPASQL is an extension of SQL allowing the execution of SPARQL queries within SQL statements, typically by treating them as sub-queries or function clauses. Variables in a SPARQL query are treated like any variable in an SQL query, making the two languages interchangeable to query the same database.

SPASQL applies an automatic direct mapping approach, along with the on-demand query processing. No query rewriting process is needed, the SQL interpreter is extended to support both SQL and SPARQL. The support of SPARQL is limited to SELECT clauses with no variable in the predicate position of triple patterns. Results are returned like usual MySQL result sets.

Sustainability: SPASQL was designed in the project FeDeRate for Drug Research[40] and remained in an early prototype status . No update was reported since 2008.

4.2.10. SquirrelRDF

With SquirrelRDF[41], Seaborne et al. [48] propose a tool to apply direct mapping rules to an input database or an LDAP directory, and to answer SPARQL queries through an on-demand query rewriting process. Mapping principles roughly follow [8]. The mapping description file (in RDF/Turtle) can be generated automatically from the database. Note that we could not find a definition of the mapping file vocabulary on the internet nor in the downloaded archives. The mapping may be customized in order to use an existing target vocabulary, nevertheless the customization is rather limited, and the mapping essentially remains driven by direct mapping rules. Jena rules may be used to produce additional knowledge in the perspective of more domain-driven mappings.

The data can be retrieved using SPARQL SELECT, ASK and CONSTRUCT query forms issued to the Jena SPARQL API. A CLI tool is available to test whether the configuration is properly set. A limited

[39] http://www.w3.org/wiki/SPASQL

[40] http://www.w3.org/2004/10/04-pharmaFederate/

[41] http://jena.sourceforge.net/SquirrelRDF/

demo SPARQL endpoint supports the SELECT query form. Whatever the access method used, variables are not allowed in the predicate position of triple patterns. An originality of SquirrelRDF is that it can query multiple databases simultaneously (n tables from m databases).

Sustainability: SquirrelRDF has not been updated since 2006. It was delivered as part of the Jena framework distribution in 2006, however it is no longer part of it since Jena[42] moved to the Apache foundation.

4.2.11. Triplify

The goal of Triplify[43] is to enable popular Web applications (like CMS or blog applications) to publish the content of their relational database as RDF Linked Data or JSON. Given that many instances of such web applications are deployed over the internet, helping them to quickly expose their database is expected to result in a boost of the Semantic Web adoption [4].

Triplify is a simple but effective approach to be used as a lightweight easy-to-learn plug-in for existing web applications. It is based on the mapping of HTTP URI requests onto queries to the relational database. Compared to the direct mapping approach, Triplify takes the problem the other way round: it first focuses on the relational data that actually matters and the queries that should be needed, instead of translating the whole relational schema into an ad-hoc ontology. Mappings are implemented as SQL statements embedded in PHP scripts, therefore any SQL construct or aggregation function of the back-end may be used. Transformation functions written in PHP may in turn be applied to the data returned by the SQL queries. Authors argue that "Triplify facilitates the creation of custom-tailored search engines targeted at certain niches, e.g. searching for specific content in various blogs, wikis, or forums" [4].

Triplify provides the on-demand and data materialisation implementations. In the on-demand mode, data is retrieved as RDF Linked Data or JSON (both through HTTP GET), but no query rewriting process is supported.

Some Triplify modules specifically address the provenance metadata generation (using the Provenance Vocabulary[44]), as well as the support of the *Link Data Update Logs* by interpreting URIs that contain the update keyword, and mapping them to appropriate queries to update the database.

Triplify is aimed at small to medium Web applications (i.e. less than 100MB database content). However, it supports the caching of the triplification results and can hence be used with large Web applications (160GB data for OpenStreetMap). It has been adapted to several popular web applications (WordPress, Joomla, osCommerce, etc.).

Sustainability: Triplify is provided under the LGPL v2 licence. The last version 0.8 was published in March 2010. However it seems to have a large and active user communities. Triplify is maintained by the Agile Knowledge Engineering and Semantic Web[45].

4.3. Summary

Table 2 summarizes the characteristics of the tools studied in this section along the three axis defined in section 2 and the mapping features listed in Table 1. The two columns under the Characteristics category specify whether each tool comes with a specific mapping language and/or an implementation of a transformation engine.

5. Synthesis

The previous section shows a large diversity of approaches among the tools designed to translate relational databases into RDF. Overall, although the various approaches are driven by various motivations, three steps cannot be circumvented: the mapping description (direct vs. domain semantics-driven), the mapping implementation (data materialisation or on-demand) and the data retrieval (query-based access or linked data). These steps have been described in sections 2.1, 2.2 and 2.3. Figure 1 depicts their sequential composition as well as the options applicable at each step. The figure reads from top to bottom, following the chronological order involved when setting up the translation process.

[42] http://jena.sourceforge.net/SquirrelRDF/

[43] http://triplify.org/

[44] http://sourceforge.net/apps/mediawiki/trdf/index.php?title=Provenance_Vocabulary

[45] http://aksw.org/About.html

Table 2

Synthetic view of tools features. The "X" cross means supported, an empty cell means unsupported. Question marks mean that available information did not permit to figure out if the feature is supported or not.

		Characteristics		Mapping description			Mapping implementation		Data retrieval		Features of mapping description													
	Tools	Mapping language	Transformation engine	Direct mapping	Augmented Direct mapping	Domain Semantics-Driven mapping	Data materialisation	On-demand mapping	Query-based access	Linked data	User defined unique Ids	Logical table	Column selection	Column renaming	Select conditions	Vocabulary reuse	1 table to n classes	many-to-many to simple triples	Blank nodes	Data types	Data transformation	Named graphs	User-defined namespaces	Static metadata
Non R2RML	D2RQ	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X
	Datalift		X	X		X	X		X	X	X			X		X	X	X	X	X		X	X	X
	DB2OWL				X		n.a		n.a				X	?				X		?			?	?
	METAmorphoses	X	X			X	X		n.a			X	X	X	X	X	X	X		X			X	X
	R2O/ODEMapster	X	X			X	X	X	X		X	X	X	X	X	X	X	X	X	X	X		X	
	RDBToOnto				X		X		n.a		X	?	X	X	X		X	X	?	X	?	?	X	?
	Relational.OWL				X		X		n.a			X	X						X	?			?	
	SPASQL		X	X				X	X											?		?	?	?
	SquirrelRDF		X	X				X	X				X	X		X			X	?			?	?
	Triplify		X			X	X	X		X	X	X	X	X	X	X	X	X	X	?	X		X	X
R2RML	DB2Triples		X	X		X	X		n.a		X	X	X	X	X	X	X	X	X	X		X	X	X
	Morph-RDB		X			X	X	X	X		X	X	X	X	X	X	X	X		X	?	X	X	
	Oracle 12c		X			X		X	X		?	?	?	?	?	?	?	?	?	?	?	?	?	?
	RDF-RDB2RDF		X	X		X	X		n.a		X	X	X	X	X	X	X	?	X	?			X	X
	Virtuoso RDF Views	X	X			X		X	X	X	X	X	X	X	X	X	X	X	X	?	?	?	X	?
	Ultrawrap		X	X		X		X	X	X	X	X	X	X	X	X	X	X	X	X	?	X	X	X
	XSPARQL	X	X			X	X		X		X	X	X	X	X	X	X	X	X	X	X		X	X

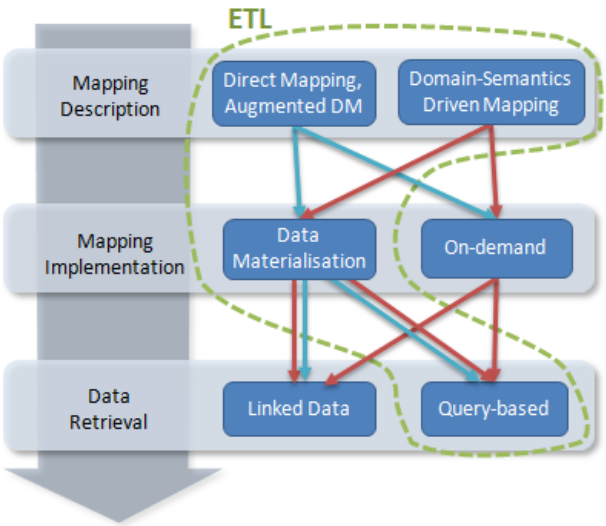

Fig. 1. Steps of the RDB-to-RDF process. From top to bottom, the figure depicts the steps needed to set up the translation process. Blue paths start with a direct mapping description, red paths start with a domain semantics-driven mapping description

The insight gained from this study suggests that the mapping description method (top layer in Figure 1) splits existing approaches into two families with regards to the overall RDB-to-RDF process: the direct mapping family is denoted by the blue arrows on Figure 1, and the domain semantics-driven mapping family is denoted by the red arrows. In section 5.1 we highlight both families by examining the mapping steps orchestration strategies, and we underline the relationship between the initial motivation and the way the tools were designed. Then, in section 5.2, we underline that the expressiveness of existing mapping languages is not always sufficient to produce semantically rich data and make it usable, interoperable and linkable. We therefore briefly present various strategies investigated in the literature to produce additional knowledge. Finally, we discuss the applicability of R2RML to the wide scope of RDB to RDF translation needs.

5.1. Direct Mapping vs. Domain Semantics-Driven Mapping

5.1.1. Direct Mapping and Augmented Direct Mapping

The direct mapping approach transforms the content of a relational database into RDF by reflecting the relational schema in an ad-hoc ontology. Nevertheless, it is hardly a goal in itself and it is generally complemented by further steps depending on the work motivation. The initial motivation of tools applying the Direct Mapping method is generally either to quickly expose relational data on the web, or to learn ontologies from a relational databases.

Direct Mapping and On-demand Implementation

SquirrelRDF, SPASQL and Ultrawrap fall in the first category (although Ultrawrap now supports domain semantics-driven mapping with R2RML, but its primary goal was the direct mapping implementation). Their point is to address the semantic web chicken-and-egg dilemma mentioned in the introduction: they enable the publication of relational data as RDF, leaving the semantic interoperability concerns for a later step. All three provide an on-demand mapping implementation, thus focusing on the efficient implementation of a query-rewriting process. Furthermore, SPASQL and Ultrawrap rely on the optimization engine of the relational backend to optimize rewritten SQL queries.

Ontology Learning, Augmented Direct Mapping and Data Materialisation

Ontology learning approaches focus on the engineering of ontologies by extracting classes and properties from relational schemas and data. The scope varies from the search for specific database design patterns such as many-to-many relations or subsumption relations, to data analysis techniques that help discover or refine ontological classes and properties. Often this is accompanied by semi-automatic methods that iteratively extend the ontology with classes and properties learnt from the database, or suggest probable matches with existing ontologies. Various such strategies can be found in [50] and [53].

All the ontology learning tools studied in this paper (DB2OWL, RDBToOnto, Relational.OWL) apply the augmented direct mapping description, that Spanos et al. [53] name *creation of a domain-specific ontology with database reverse engineering*. This choice seems natural since their goal is not to align the database on existing ontologies but to discover ontological knowledge. This is corroborated by Sequeda et al.'s survey of direct mapping approaches [50]: all of the seven reviewed papers mainly address ontology learning through the augmented direct mapping principles.

Most ontology learning approaches implement data materialisation. This is the case of RDBToOnto and Relational.OWL (DB2OWL does not provide any implementation) as well as the seven papers reviewed by Sequeda et al. [50]. This choice does not seem obvious

though, as an on-demand implementation can apply too, as illustrated by Ultrawrap. Two reasons mainly explain this choice. (i) As a pragmatic reason, ontology learning approaches need to demonstrate their feasibility, keeping the development effort as low as possible. In this regard, implementing the data materialisation method is more straight-forward than the on-demand mapping. (ii) Some ontology learning methods entail schema and data analysis possibly requiring many requests to the RDF dataset, e.g. statistical analysis of occurrences of values in a table, which may be very inefficient in an on-demand implementation.

5.1.2. Domain Semantics-Driven Mapping

Tools based on domain semantics-driven mapping generally result from two motivations, the implementation of an efficient transformation engine or the definition of a generic mapping language (or eventually both).

Asio SBRD, D2R server, Triplify, Morph-RDB, Oracle, Virtuoso and Ultrawrap target the development of production-class query engines able to handle a large number of concurrent, potentially complex requests while ensuring acceptable performances. Those tools apply the on-demand mapping implementation: they investigate methods to efficiently rewrite a query on the target data model (either SPARQL or linked data HTTP dereferencing) into an SQL query, and efficiently execute the translated query (query optimization and query planning).

Such tools may come with the definition of an expressive generic mapping language (D2RQ, Virtuoso), unless they rely on an existing language such as R2RML. The mapping language is even the prior focus of R2O/ODEMapster and METAmorphoses that both implement data materialisation. To some varying extend, mapping languages rely on a mix of specific constructors and embedded SQL snippets. Besides, Asio, Datalift and Virtuoso can be classified in a data integration framework category: their focus is not only to translate relational data into RDF, but also to query this data along with other data sources. Datalift applies data materialisation to merge heterogeneous sources as multiple graphs in a single RDF repository, while Asio and Virtuoso apply on-demand mapping.

5.1.3. Data Retrieval Methods

Query-based retrieval is the most commonly implemented method within the tools we have studied, as evidenced by the "Query-based access" column in Table 2. Five tools support the Linked Data method (D2RQ, Datalift, Triplify, Virtuoso, Ultrawrap). Triplify is the only one to support exclusively the Linked Data method.

5.1.4. Extract-Transform-Load (ETL)

The Extract-Transform-Load (ETL) expression is used in the literature to denote a 3-step approach: define a mapping, materialise the RDF data based on that mapping, and load the RDF data into a triple store to query it, typically through a SPARQL endpoint. The green dotted line in Figure 1 denotes this whole process. It shows that any of the direct or domain semantics-driven mappings can apply. Strictly speaking, and as there is no formal definition of the ETL approach, the Linked Data retrieval method could also be included. In practice though, ETL generally refers to query-based retrieval, and more precisely the use of a SPARQL endpoint.

5.2. A posteriori Production of Semantically Rich Data

Producing RDF data with sufficient semantics, in order to make it usable, interoperable and linkable, is a critical concern in most RDB-to-RDF tools. Experimentations that use direct mapping-based tools often underline the weaknesses of this method: the reuse of the RDF data produced is not easy due to the lack of formal semantic reference. Hence, its interoperability with other datasets requires to lift the data to a higher level of semantic formalisation. Furthermore, even in the case of a domain semantics-driven mapping, the production of sufficiently semantically rich data cannot always be achieved by the expressiveness of existing mapping languages. In other words, the alignment of RDF data with existing ontologies has to be considered carefully in any case. Consequently, various strategies are investigated in the literature to produce additional semantics, a posteriori. Below we briefly present most common ones.

Aligning multiple levels of ontologies

Several RDB-to-RDF projects propose to enhance the direct mapping by aligning the local ontology with higher levels of abstraction formalized by domain and application ontologies. In this case, the domain semantics-mapping is not performed beforehand as in domain semantics-driven mapping tools, but involves the later alignment of ontologies.

Asio SBRD, DataLift, as well as a DB2OWL-based approach [30], propose comparable methods using two distinct levels of ontologies. In a first step, the direct mapping method is applied to create a local ontology

that reflects the structure of the relational database. In a second step, the local ontology is manually aligned with an ontology that models some part of the domain which the relational database refers to. A query engine translates a query to the domain ontology into a query to the local ontology. The query to the local ontology is in turn translated into an SQL query to the relational database.

Alternatively, the alignment of the local ontology with a domain ontology may be achieved by using SPARQL CONSTRUCT or INSERT queries: given a graph pattern expressed with local ontology classes and properties, they produce new RDF data using classes and properties of the domain ontology. De Laborda and Conrad [23] describe a use case in which they have applied this method with Relational.OWL. This turns out to be equivalent to the execution of rules in a rule engine (see below). The difference, here, is that the rules description language is SPARQL.

Hu and Qu [37] propose to apply data mining techniques to automatically disco ver simple mappings between the relational entities (tables, columns, integrity constraints) and the classes and properties of an existing ontology. Relational and ontological entities are compared using distance measures such as the TF-IDF (*Term Frequency-Inverse Document Frequency*), in addition to the computation of the confidence in a mapping and a validation phase. In this sense, it can be seen as an automatic, domain semantics-driven approach. However, for this method to perform efficiently, the similarity between the relational schema and the ontology should be high, unless human users provide initial reference mappings.

Besides the aforementioned generic methods, some domain-specific projects propose alternative methods. For instance, Green et al. [32] describe a demonstrator that relies on existing tools to allow for the "spatial attribution" of RDF data sources, for the predictive modelling of diffuse water pollution. The federation of different data sources lies on a 3-layer ontology stack. The ontologies at the first level (called data ontologies) are used to map each data source to classes of domain ontologies (at the second level) written by domain experts. At the top, the application ontology links the domain ontologies together with the addition of application-specific information. Much of the application ontology is created manually, as it requires domain knowledge of the scenario being modelled. D2RQ is used to map data sources to the data ontologies, and is extended to include spatial operators provided by the Oracle Spatial database.

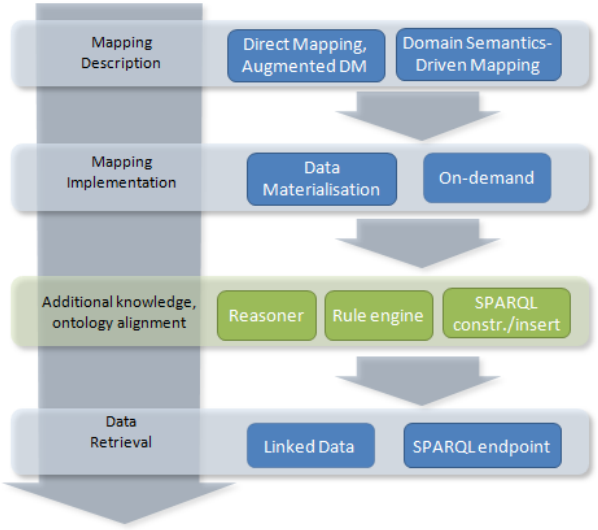

Fig. 2. Semantic enrichment. The green layer depicts strategies aimed at a posteriori production of additional semantic data

Using rules to infer additional knowledge

Rule engines or reasoners are an alternative way to enrich the semantics of RDF data. This option is depicted by the green layer in Figure 2. Such techniques can easily be used in conjunction with data materialisation in which case the whole graph is available at once. The NeuroLOG project [31] illustrates this method: it uses METAmorphoses to create an RDF materialisation of federated neuro-imaging relational databases. Some relational columns help infer whether resources belong to specific classes of the application ontology (1-table-to-n-classes property in table 1). As the METAmorphes mapping language is not capable of describing such categorization patterns, they are described as SWRL rules executed by the CORESE [17] semantic search engine. By raising the semantics of the data, the process improves the possibilities of further application-specific reasoning.

Rule engines and reasoners can also apply on RDF data translated on-demand, although the execution of complex deduction rules, or the application of complex entailments, may generate a high volume of queries to the database. In the worst case, the process may require to translate the whole relational data to complete. Among the tools studied in this paper, several support rules: SWRL rules for Asio, Silk module in DataLift, Jena Rules in SquirrelRDF, and proprietary RDF-to-RDF deduction rules in XSPARQL.

5.3. R2RML or not?

Today, R2RML seems to be inescapable when it comes to transform relational data into RDF. It results from the study of previous initiatives, it takes advantage of their experience and encompasses most of their expressiveness. To our knowledge, one graphical editor currently proposes to help users write R2RML mappings [49]. However, along with the wider adoption of the language, it is likely that other initiatives will show up. Additionally, major actors born before R2RML now comply with it (D2R server, Virtuoso, Ultrawrap). Consequently, anyone looking for a mature tool to perform RDB-to-RDF translation will most likely choose an R2RML compliant initiative.

Nevertheless, some situations suggest that R2RML cannot be considered as a "catch-all" mapping language. Firstly, direct mapping based approaches do not necessarily require a mapping language: direct mapping rules can easily be encoded in a program without needing to interpret a mapping language. Secondly, certain approaches (such as ontology learning) deal with complex patterns that cannot be expressed in R2RML. For instance, RDBToOnto [16] analyses data redundancy to discover categorization patterns. Hu and Qu [37] apply data mining techniques to automatically discover mappings between the database and an existing ontology. Lastly, R2RML does not provide data manipulation functions but relies on the capabilities of the relational backend instead. To address those situations, a further version of R2RML could, for instance, provide language extension points to enable the invocation of domain specific functions provided as a software plug-in.

6. Conclusion and Open Questions

Many techniques and tools have been proposed over the last years to enable the publication of relational data on the web in RDF. RDB-to-RDF methods are one of the keys to populate the web of data by unlocking the huge amount of data stored in relational databases. In this paper, we have described three axes to categorize such RDB-to-RDF approaches. We have proposed a detailed review of seventeen RDB-to-RDF tools, either supporting the R2RML W3C recommendation or providing their own mapping language. The proposed categorization helped us identify commonalities and differences between existing tools, underline the relationship between the initial motivation and the way the tools were designed, and discuss the orchestration of the solutions applicable at each step of the RDB-to-RDF translation process. Since producing RDF data with sufficiently rich semantics is often important in order to make the data usable, interoperable and linkable, we have also briefly presented various strategies investigated in the literature to enrich data semantics. Finally, we have shown that, although R2RML is a promising language, it may not apply to all situations in the wide scope of RDB to RDF translation needs, leaving space for future extensions.

Beyond RDB-to-RDF methods, a wide range of other initiatives target the publication of non-RDF data sources in RDF, such as academic tools (XSPARQL supports XML sources, Datalift supports CSV and XML sources), ontology-based access to data streamed from sensor webs [15], extension of R2RML to support XML and JSON data sources [25], wrappers of popular web site APIs (e.g. *flickrdf* library) or miscellaneous format-specific converters and "RDFizers". Whatever the method though, the result is a read-only web of data. And similarly to the tremendous change of paradigm that lead to the Web 2.0 where the user contribution plays a major role, the web of data will necessarily have to shift to a write-enabled model. In the case of relational databases, although the expressiveness of RDB-to-RDF mappings is a key to rich semantic alignment, it also often results in one-way transformation mappings (one-way mappings cannot be used to revert RDF data to the original relational data). Consequently, it will be necessarily to find a compromise between the expressiveness of RDB-to-RDF mapping languages and the need for updating relational data using protocols of the semantic web. The R3M mapping language is an example of the search for such a compromise [35].

Berners-Lee et al. [10] investigate some of the issues and challenges related to a read-write web of data. They underline that creating, updating and deleting RDF data should only be made possible in a secure, reliable, trustworthy and scalable way. This questions the way to authenticate people and programs in the web of data, to securely assign authorizations, to track the changes made, etc. As to the trustworthiness question, in a global data space made of heterogeneous data sources integrated together, it will become crucial to explain query results, or to assess the confidence a user can have in query results. To this end, applications exposing non-RDF data sources into RDF will have to ensure the preservation of provenance information, allowing to track a piece of data back to its original data source.

References

[1] H. Akil, M. E. Martone, and D. C. Van Essen. Challenges and opportunities in mining neuroscience data. *Science*, 331(6018): 708–712, 2011. ISSN 0036-8075, 1095-9203.

[2] M. Arenas, A. Bertails, E. Prud'hommeaux, and J. Sequeda. A direct mapping of relational data to RDF, Sept. 2012. URL http://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/.

[3] M. Ashok. W3C RDB2RDF incubator group report, 2009. URL http://www.w3.org/2005/Incubator/rdb2rdf/XGR-rdb2rdf-20090126/.

[4] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: light-weight linked data publication from relational databases. In *Proceedings of the 18th International conference on World Wide Web*, pages 621–630, Madrid, Spain, 2009.

[5] S. Auer, L. Feigenbaum, D. Miranker, A. Fogarolli, and J. Sequeda. Use cases and requirements for mapping relational databases to RDF, June 2010. URL http://www.w3.org/TR/2010/WD-rdb2rdf-ucr-20100608/.

[6] J. Barrasa, O. Corcho, and A. Gómez-Pérez. Fund finder: A case study of database-to-ontology mapping. In *Semantic Integration Workshop, in Proceedings of the 2nd International Semantic Web Conference (ISWC 2003)*, 2003.

[7] J. Barrasa, O. Corcho, and A. Gómez-Pérez. R2O, an extensible and semantically based database-to-ontology mapping language. In *Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB 2004)*, volume 3372, Toronto, Canada, 2004. Springer-Verlag. ISBN 978-3-540-24576-6.

[8] T. Berners-Lee. Relational databases and the semantic web, in design issues of the WWW, 1998. URL http://www.w3.org/DesignIssues/RDB-RDF.html.

[9] T. Berners-Lee. Linked data, in design issues of the WWW, 2006. URL http://www.w3.org/DesignIssues/LinkedData.html.

[10] T. Berners-Lee, R. Cyganiak, M. Hausenblas, J. Presbrey, O. Seneviratne, and O.-E. Ureche. Realising a read-write web of data, 2009. URL http://web.mit.edu/presbrey/Public/rw-wod.pdf.

[11] C. Bizer. D2R MAP - a database to RDF mapping language. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*, Budapest, Hungary, 2003.

[12] C. Bizer and R. Cyganiak. D2R server - publishing relational databases on the semantic web. In *Proceeding of the 5th International Semantic Web Conference (ISWC 2006)*, 2006.

[13] C. Bizer and A. Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, 2004.

[14] A. Burgun, O. Bodenreider, et al. Accessing and integrating data and knowledge for biomedical research. *Yearbook of Medical Informatics*, 47 Suppl. 1:91–101, 2008.

[15] J.-P. Calbimonte, H. Jeung, O. Corcho, and K. Aberer. Enabling query technologies for the semantic sensor web. *International Journal on Semantic Web and Information Systems*, 8 (1):43–63, 2012. ISSN 1552-6283, 1552-6291.

[16] F. Cerbah. Learning highly structured semantic repositories from relational databases: The RDBToOnto tool. In *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*, pages 777–781, Athens, GA, USA, 2008. ISBN 3-540-68233-3 978-3-540-68233-2.

[17] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the semantic web with corese search engine. In *Proceedings of 16th European Conference on Artificial Intelligence*, volume 16, page 705, Valencia, Spain, 2004.

[18] N. Cullot, R. Ghawi, and K. Yetongnon. DB2OWL : A tool for automatic database-to-ontology mapping. In *Proceedings of the 15th Italian Symposium on Advanced Database Systems*, pages 491–494, Torre Canne, Fasano, BR, Italy, 2007.

[19] R. Cyganiak, C. Bizer, O. Maresch, and C. Becker. The D2RQ mapping language v0.8, 2012. URL http://d2rq.org/d2rq-language.

[20] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language, Sept. 2012. URL http://www.w3.org/TR/2012/REC-r2rml-20120927/.

[21] C. P. De Laborda and S. Conrad. Querying relational databases with RDQL. In *Proceeding of the Berliner XML Tage 2005*, pages 161–172. Citeseer, 2005.

[22] C. P. De Laborda and S. Conrad. Relational.OWL: a data and schema representation format based on OWL. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling*, volume 43, pages 89–96, Newcastle, Australia, 2005.

[23] C. P. De Laborda and S. Conrad. Database to semantic web mapping using RDF query languages. *Conceptual Modeling-ER 2006*, pages 241–254, 2006.

[24] M. Dean. Suggestions for semantic web interfaces to relational databases. In *W3C Workshop on RDF Access to Relational Databases*, Cambridge, MA, USA, 2007. URL http://www.w3.org/2007/03/RdfRDB/papers/dean.html.

[25] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. RML: a generic language for integrated RDF mappings of heterogeneous data. In *Proceedings of the 7th Workshop on Linked Data on the Web (LDOW2014)*, Seoul, Korea, 2014.

[26] C. Dolbear and J. Goodwin. Position paper on expressing relational data as RDF. In *W3C Workshop on RDF Access to Relational Databases*, 2007. URL http://www.w3.org/2007/03/RdfRDB/papers/dolbear.pdf.

[27] O. Erling. Requirements for relational-to-RDF mapping, blog additional, 2008. URL http://www.openlinksw.com/weblog/oerling/index.vspx?page=&id=1434.

[28] O. Erling. Requirements for relational to RDF mapping, 2008. URL http://www.w3.org/wiki/Rdb2RdfXG/ReqForMappingByOErling.

[29] L. Feigenbaum, G. Todd Williams, K. Grant Clark, and E. Torres. SPARQL 1.1 protocol, Mar. 2013. URL http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/.

[30] R. Ghawi and N. Cullot. Database-to-ontology mapping generation for semantic interoperability. In *Third International Workshop on Database Interoperability (InterDB 2007), in conjunction with VLDB 2007*, Vienna, Austria, 2007.

[31] B. Gibaud, G. Kassel, M. Dojat, B. Batrancourt, F. Michel, A. Gaignard, and J. Montagnat. NeuroLOG: sharing neuroimaging data using an ontology-based federated approach. In *Proceedings of the AMIA Annual Symposium*, volume 2011, page 472, Washington DC, USA, 2011.

[32] J. Green, C. Dolbear, G. Hart, J. Goodwin, and P. Engelbrecht. Creating a semantic integration system using spatial data. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, pages 26–30, Karlshue, Germany, 2008.

[33] S. Harris and A. Seaborne. SPARQL 1.1 query language, Mar. 2013. URL http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[34] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. *Communications of the ACM*, 50(5):94–101, 2007. ISSN 0001-0782. .

[35] M. Hert, G. Reif, and H. C. Gall. Updating relational data via SPARQL/update. In *Proceedings of the EDBT/ICDT Workshops 2010*, page 24, 2010.

[36] M. Hert, G. Reif, and H. C. Gall. A comparison of RDB-to-RDF mapping languages. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 25–32, Graz, Austria, 2011. ACM.

[37] W. Hu and Y. Qu. Discovering simple mappings between relational database schemas and ontologies. *The Semantic Web*, pages 225–238, 2007.

[38] N. Lopes, S. Bischof, S. Decker, and A. Polleres. On the semantics of heterogeneous querying of relational, XML and RDF data with XSPARQL. In *Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)*, Lisbon, Portugal, 2011.

[39] C. Ogbuji. SPARQL 1.1 graph store HTTP protocol, Mar. 2013. URL http://www.w3.org/TR/2013/REC-sparql11-http-rdf-update-20130321/.

[40] Oracle. Semantic technologies in oracle database 11g release 2: Capabilities, interfaces, performance, 2010. URL http://download.oracle.com/otndocs/tech/semantic_web/pdf/2010_ora_semtech_capintper.pdf.

[41] Oracle. Oracle spatial and graph 12c, RDF semantic graph, 2013. URL http://download.oracle.com/otndocs/products/semantic_tech/pdf/12c/rdfsemanticgraph_12c_fo.pdf.

[42] F. Priyatna. *RDF-based access to multiple relational data sources*. PhD thesis, Universidad Politecnica de Madrid, Madrid, Spain, 2009.

[43] F. Priyatna, O. Corcho, and J. Sequeda. Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. In *Proceeding of the World Wide Web Conference 2014*, Seoul, Corea, 2014.

[44] E. Prud'hommeaux. SPASQL: SPARQL support in MySQL, 2007. URL http://www.w3.org/2005/05/22-SPARQL-MySQL/XTech.

[45] L. Rhys. Dereferencing HTTP URIs - draft tag finding 31 august 2007, 2007. URL http://www.w3.org/2001/tag/doc/httpRange-14/2007-08-31/HttpRange-14.html.

[46] A. Ruttenberg, T. Clark, W. Bug, M. Samwald, O. Bodenreider, H. Chen, D. Doherty, K. Forsberg, Y. Gao, V. Kashyap, et al. Advancing translational research with the semantic web. *BMC bioinformatics*, 8(Suppl 3):S2, 2007.

[47] S. Sahoo, W. Halb, S. Hellman, K. Idehen, T. Thibodeau, S. Auer, J. Sequeda, and A. Ezzat. A survey of current approaches for mapping of relational databases to RDF, 2009. URL http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport_01082009.pdf.

[48] A. Seaborne, D. Steer, and S. Williams. RDF-SQL (SquirrelRDF). In *W3C Workshop on RDF Access to Relational Databases*, 2007. URL http://www.w3.org/2007/03/RdfRDB/papers/seaborne.html.

[49] K. Sengupta, P. Haase, M. Schmidt, and P. Hitzler. Editing R2RML mappings made easy. In *12th International Semantic Web Conference (posters and demos)*, Sydney, Australia, 2013.

[50] J. Sequeda, S. H. Tirmizi, O. Corcho, and D. P. Miranker. Survey of directly mapping SQL databases to the semantic web. *Knowledge Eng. Review*, 26(4):445–486, 2011.

[51] J. F. Sequeda and D. P. Miranker. Ultrawrap: SPARQL execution on relational data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 22:19–39, Oct. 2013. ISSN 15708268.

[52] J. F. Sequeda, R. Depena, and D. P. Miranker. Ultrawrap: Using sql views for rdb2rdf. In *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, Chantilly, VA, USA, 2009.

[53] D.-E. Spanos, P. Stavrou, and N. Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web Journal*, 3(2):169–209, 2012.

[54] M. Svihla and I. Jelinek. Two layer mapping from database to RDF. In *Proceedings of Electronic Computers and Informatics (ECI)*, 2004.

[55] S. Tirmizi, J. Sequeda, and D. Miranker. Translating sql applications to the semantic web. In *Proceedings of the 19th international conference on Database and Expert Systems Applications (DEXA'08)*, pages 450–464, 2008.