Undefined 0 (2010) 1–17
IOS Press

Bringing Relational Databases into the Semantic Web: A Survey

Dimitrios-Emmanuel Spanos [a,*], Periklis Stavrou [a] and Nikolas Mitrou [a]
[a] *National Technical University of Athens, School of Electrical & Computer Engineering, 9, Heroon Polytechneiou str., 15773 Zografou, Athens, Greece*
E-mail: {dspanos,pstavrou}@cn.ntua.gr,mitrou@cs.ntua.gr

Abstract. Relational databases are considered one of the most popular storage solutions for all kinds of data and they have been recognized as a key factor in generating huge amounts of data for Semantic Web applications. Ontologies, on the other hand, are one of the key concepts and main vehicle of knowledge in the Semantic Web research area. The problem of bridging the gap between relational databases and ontologies has attracted the interest of the Semantic Web community, even from the early years of its existence and is commonly referred to as the database-to-ontology mapping problem. However, this term has been used interchangeably for referring to two distinct problems: namely, the creation of an ontology from an existing database instance and the discovery of mappings between an existing database instance and an existing ontology. In this paper, we clearly define these two problems and present the motivation and benefits for both of them. We attempt to gather the most notable approaches proposed so far in the literature, present them concisely in tabular format and group them under a classification scheme. We finally explore the perspectives and future research steps for a seamless and meaningful integration of databases into the Semantic Web.

Keywords: Relational Database, Ontology, Mapping, OWL, Survey

1. Introduction

Over the last decade or so, the Semantic Web (SW) has grown from a futuristic vision into a multidisciplinary research field, gathering researchers and gaining expertise from other scientific fields, such as artificial intelligence, software engineering and computer networks, to name a few. Hence, it comes as no surprise that efforts to bring database theory and database systems in general, to the Semantic Web universe date almost as old as its conception.

Initially, database systems were considered by the Semantic Web community as an excellent means for efficient ontology and SW data storage, because of their largely known performance benefits. This consideration has led to the development and production of several database systems, especially optimized for the storage, maintenance and querying of SW data. Such systems are informally known as *triple stores* [13]. Then, a different type of interaction between databases and ontologies was considered, following a different motivation and the exact opposite route: bringing the contents of a database into the Semantic Web. In fact, the case of relational databases has been the most interesting so far, given their popularity and the significant challenges arising from their underlying logical model. This issue is also known as the *relational database to ontology mapping* problem [42] and is the subject of this paper. Alternatively, the acronym RDB2RDF, coined by the homonymous W3C working group[1], is also used to refer to the same problem.

The main motivation that drove the consideration of the RDB2RDF problem was the need to semantically annotate dynamic Web sites, the contents of which are retrieved by relational database records [72]. In the road though, it became evident that a solution to the

*Corresponding author. E-mail: dspanos@cn.ntua.gr.

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

RDB2RDF problem would help create a critical mass of SW data in order to overcome the infamous chicken-and-egg problem [36], that has been recognized as one of the factors hindering the realization of the Semantic Web vision [43]. Moreover, additional benefits that a solution to the mapping problem would incur, such as contribution to the long-standing data integration problem [47], brought even more attention to it.

An interesting aspect of the database to ontology mapping issue is the fact that, being loosely defined, the term is often used in the literature to describe different problems, thus creating mild confusion on the exact nature of the problem and related solutions. To be more precise, the above term has been used so far to refer to the creation of an ontology expressed in some knowledge representation language (e.g. RDFS, OWL) from a given relational database, the export of a database's contents in RDF and the discovery of correspondences between a specific ontology and a given database. We attempt to clearly define and distinguish between these, closely related but distinct, problems and classify accordingly the numerous approaches proposed in the literature.

The rest of the paper is structured as follows: Section 2 mentions the motivations and benefits of the database to ontology mapping problem, while Section 3 formally defines the problem and describes an initial solution that has served as a basis for the majority of solutions proposed. Section 4 is dedicated to the creation of a new ontology from an existing database, while Section 5 examines the problem of discovering mappings between an existing ontology and a given database. Section 6 introduces criteria and descriptive features for the approaches mentioned in order to form a complete picture of the state of the art and finally, Section 7 closes the paper discussing future steps and requirements.

2. Motivation and Benefits

The problem of mapping a database into Semantic Web did not emerge as a mere exercise of transition from one representation model to another. The motivation that drove the work of several researchers over the year and the benefits acquired by a successful solution are multifold:

Semantic annotation of dynamic web pages. Initially, the need for mapping databases to ontologies arose from the necessity to semantically annotate Web pages in order to facilitate the "upgrade" of the current Web of documents to a Web of data. For the case of static Web pages, this has been a trivial issue even before the arrival of RDFa[2]. However, dynamic Web pages, which represent the biggest part of the Web and form the so called Deep Web, retrieve their data from underlying databases and this is the case for content management systems (CMS), forums, wikis and other Web 2.0 sites [9]. In such cases, it has been argued that, instead of performing manual annotation of every generated page, it is far more practical to establish connections between the database and some ontology [72].

Ontology-based data access. Establishing correspondences between a database and an ontology greatly enhances the quality of search results and simplifies data access for the end user. Once a desired mapping is available, a user can execute queries on the ontology and retrieve data from the "linked" database. In other words, the ontology acts as a mediator between the user and the data. The result of this database and ontology coupling is also known as ontology-based data access and is supposed to lead to more meaningful queries for the user. In a similar fashion, knowledge stored in an ontology can help reformulate a SQL query to another one which better captures the intention of the user [15] or can be combined inside so called *ontology-assisted SQL queries* in appropriately modified query engines [30].

Data integration. In the last decade, ontologies have taken the place of conceptual models that used to appear in typical data integration architectures. Regardless of the exact architecture of an integration system and the number of local and global ontologies involved [73], data sources should be linked with at least one ontology. These links usually consist of mapping formulas that express a data source's components in terms of an ontology or vice versa - such mappings are called Local as View (LAV) or Global as View (GAV) respectively in the data integration literature [47]. Discovering and expressing such mappings between relational database schemas and ontologies is an integral part of a heterogeneous database integration scenario [5].

Data supply for the Semantic Web. It has been stated that one of the reasons delaying the Semantic Web realization is the lack of successful tools and applications showcasing the advantages of SW tech-

[2] In short, RDFa (`http://www.w3.org/TR/xhtml-rdfa-primer/`) is a W3C recommendation for embedding annotation in XHTML pages

nologies. The success of such tools, however, is directly correlated to the availability of a sufficiently large quantity of SW data [36]. As relational databases represent the storage medium for a vast quantity of data, it is evident that approaches exporting relational data to RDF or populating existing ontologies with instances greatly contribute to the Semantic Web success by increasing the amount of available SW data.

Ontology development. The process of manually developing an ontology is difficult, time-consuming and error-prone. Several semi-automatic ontology development methods have been proposed, extracting knowledge from free and semi-structured text documents, vocabularies and thesauri, domain experts and other sources [35]. Relational databases constitute significant structured sources of domain knowledge, allowing for, mainly automatic, ontology development methods. The fact that the design of relational databases is based on a conceptual model (e.g. UML, Extended Entity Relationship model) that is very much alike the ontology model, together with their frequent maintenance and timeliness of their data, especially in business environments [76], act as arguments in favour of using relational databases as knowledge sources for ontology development. Mapping databases to ontologies is a term often used to describe the above process.

Definition of semantics of a relational schema. Although the design of relational databases is based on a conceptual model, which is defined beforehand and then transformed to the final relational model, the initial conceptual model is often not kept alongside the logical schema. As a result, the intention implied by the logical schema is missing and this represents a major obstacle in reusing it properly, e.g. in the case of legacy systems. Defining correspondences between a database schema and an ontology provides the former with well-defined semantics and enhances the understanding of the schema, which in turn facilitates the process of re-engineering the logical schema when initial design requirements are modified and simplifies considerably the integration with other data sources [4].

3. A Naïve Solution

On the one hand, relational databases are based on the relational model, introduced by Codd [28] in 1970, while ontologies are based on RDF's graph model and Description Logics (DL) [10]. Although mapping between these two models is not straightforward, a first solution was proposed by Berners-Lee in the early years of Semantic Web [16]. Despite the fact that this solution is not ideal, since it does not cover all possible mapping cases, it has served as a foundation for a large part of related approaches having appeared since then. In this section, we will sketch this naïve approach of translating a relational database schema and its contents to an RDF graph.

To begin with though, we attempt to sketch these models. Several definitions have been given for both the relational model and ontology in computer science, depending on the intention and the view adopted by every author. We deliberately choose to give rather informal definitions as we simply intend to gain some familiarity with these models instead of giving a rigorous mathematical framework.

For the relational model, we provide an informal version of the definition from [1]:

Definition 1 *A **relational model** consists of:*

- *a set of attributes* att
- *a set of constants* dom
- *a set of relation names* $relname$
- *a function* $sort$ *associating a relation name with a finite set of attributes*

*A **relation schema** is simply a relation name. A **database schema** is a finite set* D *of relation names. A **relation instance** of a relation schema* R *is a set of tuples with* $sort(R)$*. A **database instance** of a database schema* D *is a function* I *over relation schemas* R*, such that* $I(R)$ *is a relation over relation schema* R*, for each* $R \in D$*.*

Informally, the relational model consists of a set of tables. Each table has a set of attributes or columns. Several restrictions have been incorporated over the years in the initial relational model. The most prominent ones are the primary and foreign key constraints. The *primary key* constraint indicates that each table has a column that uniquely identifies every row of the table. This column is the primary key of the table and contains distinct values. The *foreign key* constraint states that a column of a table is restricted to only contain values from the primary key of another table.

The **RDF model** is a graph-based model which is based on the concept of RDF triples. *RDF triples* consist of three components: a subject, a predicate and an object. An RDF triple is an assertion stating that a relationship, denoted by the predicate, holds between the subject and the object. An RDF graph is a set of RDF

triples. Subjects and objects of RDF triples are represented by nodes and predicates are arcs pointing from the subject to the object of the triple.

Numerous definitions have been given for the term **ontology model**, which is rather vague as it includes ontologies of several logic formalisms and representation languages. Therefore, we choose to mention the definition from [50] which is simple and general enough to describe most ontologies:

Definition 2 *An ontology is a set of primitives containing:*

- *a set of concepts C*
- *a set of relationships R, described by domain and range restrictions*
- *a taxonomy of concepts H_C with multiple inheritance*
- *a taxonomy of relationships H_R with multiple inheritance*
- *a set of axioms describing additional constraints on the ontology that allow to infer new facts from explicitly stated ones*

The most popular Web ontology language is OWL. OWL ontologies satisfy Definition 2 as they are able to contain all the primitives mentioned. The basic elements of OWL are *classes*, *properties* and *individuals*, which are members of classes. OWL properties are binary relationships and are distinguished in object and datatype properties. Object properties relate two individuals, while datatype properties relate an individual with a literal value. Hierarchies of classes and properties, property domain and range restrictions, as well as value, cardinality, existential and universal quantification restrictions on the individuals of a specific class can be defined, among others. OWL 1 defines three sublanguages: OWL Lite, OWL DL and OWL Full, in order of increasing expressiveness. OWL 2, on the other hand, defines three profiles of different expressiveness, with overlapping feature sets: OWL 2 EL, OWL 2 RL and OWL 2 QL[3].

The similarities between the relational model and, initially, RDF have led Berners-Lee to propose an informal transformation method from a given relational database instance to an RDF graph. Thus, according to [16]:

1. Every tuple of a relation R maps to an RDF node
2. Every attribute att of a relation maps to an RDF property $P(att)$
3. For every tuple $R[t]$, the value of an attribute att maps to a value of the property $P(att)$ for the node corresponding to the tuple $R[t]$.

Extensions of this approach have considered adding RDFS vocabulary descriptions, by mapping every relation R to an RDFS class $C(R)$ and adding RDF statements for declaring $C(R)$ as the type of the RDF nodes generated by relation R [13]. Furthermore, as every RDF node is identified with a URI, directives for the generation of URIs are given, summarized in Table 1.

This naïve approach is generic, since it can be applied to any relational database instance and automatic, as it does not require any human input, except for the stem URI. Thus, it allows for a "quick and dirty" exportation of relational databases to RDF. Unfortunately, this method is highly restrictive, not permitting complex mappings or supporting more expressive schema languages (e.g. OWL), while the resulting RDF graph is far from perfect, since it is essentially a copy of the database logical schema in RDF form, where domain semantics are absent. Representative example of this deficit is the mapping of relational model artifacts (e.g. tables representing a many-to-many relationship between two entities, lookup tables, surrogate keys etc.) to RDFS classes. Other notable shortcomings of this basic approach include its URI generation mechanism, where new URIs are minted even when there are existing URIs fitting for the purpose and the excessive use of literal values, which seriously degrade the quality of the RDF graph and complicate the process of linking it with other RDF graphs [22].

Nevertheless, this approach served as a starting point for numerous other relevant efforts and is often referred to in the literature as the "table-to-class, column-to-predicate" approach. Tools implementing refined versions of the naïve solution as well as efforts to extract expressive ontologies from a relational database instance are presented in the next section.

4. Ontology Creation from Existing Database

As outlined in Section 2, generating Semantic Web content from relational database instances can bring significant benefits. This has led to a large amount of work aiming at creating ontologies (in the broad sense of Definition 2) from relational database instances.

[3]For more details, see `http://www.w3.org/TR/owl2-profiles/`

Table 1
Default URI Generation Scheme

Database Element	URI Template[a]	Example
Database	$\{base\_URI\}/\{db\}/$	http://www.example.org/univ/
Relation	$\{base\_URI\}/\{db\}/\{rel\}$	http://www.example.org/univ/staff/
Attribute	$\{base\_URI\}/\{db\}/\{rel\}\#\{attr\}$	http://www.example.org/univ/staff#address
Tuple	$\{base\_URI\}/\{db\}/\{rel\}/\{pk\}.\{pkval\}$	http://www.example.org/univ/staff/id.278

[a] The name of the database is denoted db, rel refers to the name of a relation, $attr$ is the name of an attribute and pk is the name of the primary key of a relation, while $pkval$ is value of the primary key for a given tuple of a relation.

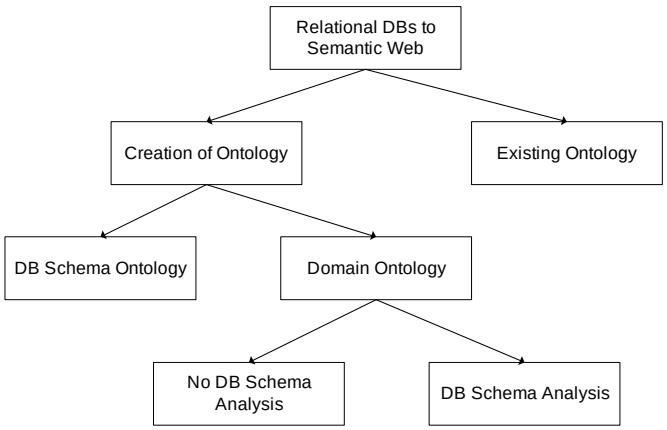

Fig. 1. Classification of approaches.

This work can roughly be classified in the following categories, also shown in Figure 1:

1. Creation of a new ontology reflecting the database schema
2. Creation of a new domain-specific ontology
   (a) Approaches not performing database schema analysis
   (b) Approaches performing database schema analysis

4.1. Generation of a database schema ontology

The first class of approaches builds ontologies which have as domain the relational model itself. Such efforts define the most important elements of the relational model and the relationships that connect them, thus forming high-level ontological representations of the database schema. Ontologies built are then naturally populated with instance data from the database instance.

The most popular and dominant work in this category is **Relational.OWL** [53], which is also reused in other approaches and tools (e.g. ROSEX [29] and DataMaster [52]). Relational.OWL embraces the "table-to-class, column-to-predicate" approach for the representation of the database schema, where the classes and properties generated from relations and attributes respectively are instances of meta-classes *Table* and *Column*. Data contained in the database is represented as instances and values of the schema representation classes and properties respectively. This immediately makes the Relational.OWL ontology OWL Full. Moreover, OWL classes denoting the concepts of primary key and database schema are defined, as well as OWL properties that tie together database schema elements (such as the *hasColumn* property connecting instances of the *Table* and *Column* classes and the *references* property describing foreign key links). The creation of the OWL Full ontology representing both the schema and the data of the database is performed automatically, without any user intervention. Performing reasoning on the Relational.OWL ontology output is essentially pointless, since ontology facts concern strictly the database schema, which the user has complete understanding of. Hence, new derived facts will not enhance domain knowledge in any way. Finally, with the addition of the RDQuery wrapper system [55] on top of Relational.OWL, it allows relational

databases to be queried through SPARQL in terms of the Relational.OWL ontology.

An approach similar to Relational.OWL that produces a new "database schema" OWL ontology is **RDB2ONT** [70]. The classes and properties defined follow the same modeling paradigm with Relational.OWL, but the attempt to model relational schema constraints as OWL constraints is erroneous. Although relational schema constraints refer to the contents of a relation, the authors translate these constraints by applying them on the schema representation OWL classes, trying to avoid meta-modeling. Avoidance of meta-modeling is successful in [45], where representation of the database contents is not considered. The output of the latter approach is an OWL-Lite ontology reflecting the database schema, which is later manually enriched with terms from a domain ontology.

Automapper [33] (now part of the commercial Asio SBRD tool) allows a relational database to be queried through SPARQL using terms from a domain ontology. It creates an OWL ontology enhanced with SWRL[4] rules to describe the relational schema and express its constraints, such as the uniqueness of the primary key or attribute datatype restrictions. Generation of the schema ontology is automatic, creating one class per table and one property per attribute. The mapping, i.e. the correspondences between the database schema and the "database schema" ontology, is expressed in terms of a mapping ontology heavily influenced by D2RQ [19], which is described in Section 4.2. SWRL rules are also employed to link the "database schema" ontology with a domain ontology. SPARQL queries on the domain ontology are then translated to SQL queries, taking into account details of the stored mapping, such as the schema-specific URI generation mechanism. A similar workflow is assumed in [31], where their automatically generated data ontology is modified accordingly to match the schema of a spatial database.

FDR2 [44] follows a different approach, building automatically an RDFS representation of the database schema and contents. Every attribute of a relation is mapped to a class, a relation corresponds to an RDF sequence container of these classes, while every pair of attributes of the same relation is mapped to a so-called virtual property. A database value is mapped as instance of its respective "column class" and a tuple is represented indirectly, with the interlinkage of values of the same row via virtual properties. This design choice leads to a certain amount of redundancy in the representation of database contents, especially in the case of relations with n attributes, where $n(n-1)$ virtual attributes are created. Correspondences of this data ontology with an existing domain ontology are defined manually.

Overall, approaches that fall in this category are mainly automatic processes that do not require any input from human user or external sources of information, as the semantics of the created ontology simply regard the logical schema of the database. Although the output of these methods is an RDFS or OWL ontology, this does not suffice for achieving semantic interoperability. Such ontologies can only serve as a uniform representation of the relational schema (one could, in fact, argue that SQL DDL statements serve the same purpose and even more conveniently), since data semantics is not examined or defined at all. Thus, in most cases, these "database schema" ontologies should be complemented with existing domain ontologies that define the semantics of the contained data [33,44,54,70].

4.2. Generation of a domain-specific ontology

Instead of creating an ontology that mirrors the database instance and adding at a later stage domain semantics, it is preferable to create directly an ontology that refers to the subject domain of the contents of the database. The expressiveness of the generated ontology largely depends on the amount of domain knowledge incorporated in the procedure. Database logical schemas often constitute an excellent source of domain knowledge, since in most cases their design is based on some conceptual model. Hence, we distinguish approaches based on whether they employ database schema analysis in order to enhance the quality of the generated ontology or not.

4.2.1. Approaches not performing schema analysis

Tools and approaches that create a new domain ontology, without analyzing the database schema to extract ontology classes and properties, are mentioned in this section. Such tools mainly rely on the "table-to-class, column-to-predicate" approach, the result of which is often refined manually from the user.

D2RQ [19] is one of the most prominent tools in the field of relational database to ontology mapping. It extracts the contents of a relational database to an RDF

[4] SWRL is an early W3C submission that allows the inclusion of rules in an OWL ontology, `http://www.w3.org/Submission/SWRL/`

graph, depending on mappings specified in a mapping language which is also expressed in RDF. D2RQ's mapping language offers great flexibility, since it allows for mapping virtually any subset of a relation in an ontology class, while the user can also control the URI generation mechanism for ontology individuals. It also offers an automatic extraction of database contents to RDF, according to the naïve solution of Section 3. D2RQ could have also been included in the group of approaches that use an existing ontology (detailed in Section 5), since mappings can be manually modified to include class and property URIs from already published vocabularies. One of the advantages of the D2RQ tool is the fact that generated RDF need not be materialized in some external file or triple store. Instead, it can serve as a programming interface and as a gateway that offers ontology-based data access to the contents of a database through either Semantic Web browsers or SPARQL endpoints. The engine that uses D2RQ mappings to translate requests from external applications to SQL queries on the relational database is called D2R Server [17].

OpenLink Virtuoso integration platform also offers an RDF view over a relational database with its **RDF Views** feature [20]. This view on the data is based once more on the classic "table-to-class, column-to-predicate" approach and expressed in a proprietary language. Definition of the mapping can be later refined to involve more complex mapping cases, as Virtuoso's mapping language has the same expressiveness to D2RQ's, allowing to assign any subset of a relation to an RDFS class and to define the pattern of the generated URIs. Hence, relational database contents can be accessed through SPARQL queries, without having to be exported and stored explicitly as RDF. One downside of both D2RQ and RDF Views is the fact that a user should familiarize himself with these mapping languages in order to perform the desired transformation of data into RDF, unless he chooses to apply the automatic naïve solution.

Triplify [9] is another notable RDF extraction tool from relational schemas. SQL queries are used to select subsets of the database schema and map them to URIs of ontology classes and properties. The mapping is stored in a PHP configuration file, the modification of which can be performed manually and consequently, URIs from existing popular vocabularies can be used. The URI generation scheme for data residing in the database resembles the basic one, depicted in Table 1. RDF triples can be either materialized or be published as Linked Data. In the latter case, a dereferenceable URI is assigned to every class and instance and the corresponding data from the database is retrieved on-the-fly. The popularity of the Triplify tool rests on the fact that it offers predefined configurations for the schemas used by popular Web applications. An additional advantage is the use of SQL query for the mapping representation, which does not require users to learn a new mapping language.

SquirrelRDF [61] is one of the first RDF extractors for relational databases. It extracts database contents as RDF triples according to mappings that are defined manually and expressed in RDF form. The vocabulary used for the expression of the mapping is lightweight, does not offer the wide range of features that D2RQ does and merely allows mapping a relation to an RDFS class and an attribute to an RDF property. **Tether** [22] is another system that translates a relational database that holds cultural heritage data to RDF by refining and solving several shortcomings of the naïve approach. However, some of the refinements proposed (e.g. to reduce the size of the RDF graph) are domain-specific and cannot be generalized for other domains.

Another tool that uses a similar to D2RQ mapping language is **METAmorphoses** [68]. Its mapping language is expressed in XML and allows the nesting of SQL queries to define the subset of the database that is mapped to an ontology class or property. Potentially, classes and properties from existing ontologies can be used. However, this tool follows a slightly more complicated approach since it also employs a template language, which lets the user select the defined mappings that wants to include in the RDF output.

To sum up, the tools of this category either resort to an automatic export of database contents to RDF following the basic approach, or give user the freedom to define manually his own mappings (or even both).

4.2.2. Approaches performing schema analysis

The tools presented in Section 4.2.1 do not take into account the database schema in order to recognize artifacts of the logical schema that correspond to ontology constructs. Analysis of the database schema can yield additional knowledge and enhance the otherwise "flat" and sparse ontology graph produced by previously mentioned methods. Approaches that perform database schema analysis present considerable variance, since they make different assumptions on the information known about the database schema (e.g. knowledge of primary and foreign keys, uniqueness of attributes, attribute dependencies) and on the avail-

ability of external sources of information (e.g. user queries, data manipulation statements, user input).

The methods of the current section have a lot in common with early *database reverse engineering* approaches. The goal of the latter is to extract the initial conceptual model – often an instance of the well-known Entity Relationship (ER) model [26] – on which the design of the logical database schema was based. Usually, reverse engineering methods make assumptions on the degree of normalization[5] of the input database schema, mine for candidate keys and inclusion dependencies between attributes in order to discover the primary and foreign keys of relations and then, interpret relations and attributes of the logical model as entities, attributes and relationships in the ER model [27,39,59]. Cardinality constraints on the participation of entities in relationships can also be extracted [3] and efforts have been made to infer hierarchies of entity types [46], in case the output of the procedure is an extended ER model. The output of a reverse engineering method can also be a more expressive object-oriented model [14,57]. The majority of these methods are semi-automatic, taking into account not only the schema but the contents of the database as well, requiring input from a human expert to validate their output and possibly, exploit user queries on the database and relevant domain thesauri [40].

The above reverse engineering approaches have served as an inspiration for the methods creating an ontology from a relational database instance. The majority of the latter methods assume as input an SQL DDL (Data Definition Language) script that contains information about the primary and foreign keys of database relations and the datatypes of relation attributes and, following mainly heuristic rules, they recognize ontology constructs and generate an OWL ontology. In most cases, database relations are classified to groups, depending on the number and concurrence of primary and foreign key attributes, and each group is mapped to a different ontology component. There is also a wealth of approaches that produce an ontology expressed in various DL languages [23,49,56], but here we mainly focus on OWL (with the exception of [67]), which is the dominant knowledge representation language in the Web. Some tools, such as ER2WO [74], ERONTO [71] or ER2OWL [32] consider as input for their algorithm an ER model and therefore, are able to produce more expressive and accurate ontologies, since ER model is an object-oriented model, not conceptually far from the OWL ontology model. However, availability of the ER model for a given database is often an unrealistic assumption, which makes the process of building an ontology from a not documented relational schema a much more challenging problem.

The sets of heuristic rules proposed by this group of methods are, to a large extent, overlapping. Broadly speaking, proposed rules can be classified to the following categories:

1. **Default rules**. These are the rules describing and extending the basic approach for the OWL paradigm, i.e. a relation maps to an OWL class, non-foreign key attributes are mapped to OWL datatype properties, foreign-key attributes are mapped to OWL object properties and a relation tuple maps to an OWL class instance.
2. **Hierarchy rules**. Such rules identify hierarchies of classes in the relational schema. According to them, two relations that have their primary keys linked with a foreign key constraint are mapped to two OWL classes, the one being a superclass of the other. These rules often conflict with *fragmentation rules*.
3. **Binary relationships rules**. These rules identify relations that are mapped to OWL object properties. The primary key of such relations is composed of foreign keys to two other relations mapped to OWL classes.
4. **Weak entity rules**. These rules identify weak entities and their corresponding owner entities. A weak entity is usually represented with a relation that has a composite primary key including a foreign key to another relation. Such relations are still mapped to OWL classes, however the semantics of the relationship connecting a weak entity with its owner are difficult to be specified.
5. **n-ary relationships rules**. These rules identify n-ary relationships, usually in cases where the primary key of a relation is composed of foreign keys to more than two other relations. Essentially, such rules are not any different from default rules, given that n-ary relationships cannot be directly expressed in OWL, hence they are mapped to an OWL class.
6. **Fragmentation rules**. These rules identify relations that have been vertically partitioned and

[5] Briefly put, normalization is the process of decomposing the relations of the database schema in order to eliminate certain types of dependencies between its attributes. The types of dependencies to be eliminated define the *normal form* of the database schema. More information can be found on database theory textbooks, e.g. [58]

correspond to the same OWL class. The identification relies on the presence of the same primary key in all relations to be merged, much like in the discovery of hierarchy case.

7. **Constraint rules**. These rules exploit additional schema constraints, which are present in SQL DDL statements. Such schema constraints include restrictions on non-nullable and unique attributes, as well as constraints regarding an attribute's datatype. These are usually mapped to respective OWL cardinality axioms on datatype properties and either global (OWL range definitions) or local universal quantification axioms (pertaining to a specific OWL class) respectively.
8. **Datatype translation rules**. These rules are essentially translation tables defining correspondences between SQL datatypes and XML Schema Types. Such correspondences are defined extensively in the SQL/XML standard.

The above rules are, in most cases, described informally [21,34,48,64,66,67] and sometimes, only through convenient examples that cannot be generalized [7,8]. Moreover, few methods yield semantically sound results, as they often contain faulty rules that misinterpret the semantics of the relational schema and in other cases, OWL constraints are misused. Two of the most complete and formalized approaches are **SQL2OWL** [2] and the work of Tirmizi *et al.* [69], in which they propose a set of Horn rules and prove that they cover all possible relation cases, distinguished according to the number of foreign keys in a relation and their concurrence with primary keys.

It should be noted that in all of the above approaches, the contents of the relational database are transformed to ontology individuals (with the exception of the R2O architecture [66] that maps only the database schema) and essentially replicated in OWL format. The only exception is **DB2OWL** [34], which stores the mapping in a R_2O document (analyzed in Section 5) and does not materialize the entire OWL ontology. DB2OWL is a module of a trivial ontology-based data integration architecture, where SPARQL queries are translated to SQL queries, according to the stored mapping and SQL results are translated back in SPARQL results form. The other exception is **Ultrawrap** [62], which builds a similar architecture on top of the mapping algorithm of Tirmizi *et al.* [69] offering an RDF view of the underlying relational database, in much the same way as D2R Server and Virtuoso RDF Views.

In contrast with reverse engineering approaches that exploit data correlations in the database instance and often employ data mining techniques, data is rarely used as a source of semantics in order to ameliorate the quality of the generated ontology. **RDBToOnto** [24] is the only tool that considers data from the database in order to extract class hierarchies. The main intuition behind RDBToOnto is the discovery of attributes that present the lowest variance of containing values. The assumption is that these attributes are probably *categorical* attributes that can split horizontally a relation R initially mapped to a class $C(R)$, separating its tuples in disjoint sets, which in turn can be mapped to disjoint subclasses of the initial $C(R)$ class.

As most of the approaches that belong to this class are based on heuristics, their efficiency is doubtful, since they cannot capture all possible database design practices and guess the original intention of the designer. Even though these methods consider only standard design practices, several problems have been identified:

Discovery of class hierarchy . Typically, hierarchies in relational databases are modeled by the presence of a primary key that is also a foreign key referring to the primary key of another table. We have already pointed out the fact that this design can also imply that a relation has been vertically partitioned and information on the same instance has been split across several relations. Still, this is not the only way to express class inheritance in the logical model: categorizing attributes in a relation may provide hints for mining subclasses.

Elicitation of highly expressive constraints So far, no methods are able to map successfully, specific OWL constructs, such as symmetric and transitive properties, since these features cannot elicited by inspection of the database schema alone. Furthermore, among the approaches reviewed, there are divergent opinions on the expression of attribute datatype restrictions. Some suggest translating these restrictions to range restrictions on the corresponding OWL datatype property, while others suggest the use of universal quantification axioms for a specific class with the *owl:allValuesFrom* construct.

Open/Closed World Assumption The desire to translate relational schema constraints to ontology axioms often leads researchers to overlook the fundamental difference on the semantics of the two formalisms. Relational databases abide by the

closed-world assumption, which states that what is not stated in a database instance is known to be false, in contrast with the open-world assumption of ontologies, where everything not explicitly stated as a fact is simply not known. Hence, while relational schema constraints define and restrict legal database instances, ontology axioms can only be used to infer new knowledge. This inconsistency between the two models has been extensively analyzed in [51,65].

5. Mapping a Database to an Existing Ontology

The approaches presented in Section 4 generate new SW data without any a priori knowledge on existing ontology schemas. In this section, we take a look at approaches that attempt to define mappings among an already developed ontology and a relational database instance. This problem is particularly challenging due to the lack of freedom of creating at will new ontology structures and the implicit constraint of not altering the given ontology. Thus, the goal is to find the best match between elements of the two formalisms.

This procedure is very hard to be automated, most approaches assume manual definition of mappings between the database instance and the ontology and only few approaches propose a semi-automatic solution, where some input from the user is required to initiate the process.

R_2O [12] is a declarative mapping language between a relational database instance and an existing ontology. It is highly expressive, as it can describe complex mapping cases that extend beyond the trivial "table-to-class, column-to-predicate" approach. Practically, it can map every possible subset of a relation to a class, allowing the combination of all relational operations. It supports description of the database schema, conditional mappings and customization of the URI generation mechanism, sharing equivalent features with the ones of D2RQ. R_2O expresses *schema-level* mappings between the two models, given that elements of the database schema are mapped to ontology classes and properties. However, its increased expressiveness is coupled with a complex representation of the mapping, rendering mapping definition difficult for the inexperienced user. The ODEMapster engine [11] exploits mappings expressed in R_2O in order to create ontology individuals from the contents of the database either in a materialized or query-driven fashion. In the latter case, data stays in the database and is not physically replicated, just like in the D2R Server, Virtuoso RDF Views and Ultrawrap tools.

Manual mappings between a database schema and a given ontology are also defined in **DartGrid** [25] and **VisAVis** [41]. DartGrid defines a typical database integration architecture, a critical component of which is the definition of mappings between each local database and a global RDFS ontology. Correspondences between components of the two models are defined via a graphical user interface and LAV mappings are generated in the background and stored in an RDF/XML format, which the authors leave unspecified. DartGrid allows ontology-based data access, by performing translation of SPARQL queries to SQL. VisAVis is also a graphical tool which allows users to define mappings between elements of an OWL ontology and a database SQL view. The mapping is expressed directly as an SQL query stored as a literal value of a special purpose datatype property added to the ontology. The use of SQL for the representation of the mapping is the main advantage of VisAVis, rendering it accessible to the average user. However, it does not provide data access through SPARQL queries.

A well elaborated semi-automatic approach is the one of **MAPONTO** [5]. This assumes that correspondences between relation attributes and properties are provided by the user. The goal of the tool is to be able to extract LAV mappings of the form $R(\bar{X}) : -\Phi(\bar{X}, \bar{Y})$, where $R(\bar{X})$ is some relation and $\Phi(\bar{X}, \bar{Y})$ is a conjunctive formula over ontology concepts with $\bar{X}$,$\bar{Y}$ being sets of variables or constants. In short, MAPONTO views the relational schema as a graph (where foreign key links are the connecting edges) and for each relation in it, tries to build a tree that will define the semantics of the relation. Foreign key links are traversed, adding visited relations to the tree, while the algorithm is aware of the various relation structures that may represent binary, n-ary relationships or weak entities. Exploiting the information on the correspondences between relation attributes and ontology properties, MAPONTO is able to encode, in a straightforward manner, the generated tree (in the relational graph) in a conjunctive formula (using ontology terms).

One of the most interesting approaches in this group of methods is the one used in **MARSON** [38]. At first, relations are categorized on groups, according to whether they represent entities or relationships, following the reverse engineering process of Chiang *et al.* [27]. Vectors of coefficients, called *virtual documents*, are then constructed for every relation and at-

tribute of the database schema. The description of a relation takes into account the description of relations connected to it through the presence of foreign keys, while the description of attributes incorporates the description of its parent relation and other relations connected to the latter as well as the attribute's datatype. Similarly, virtual documents for every class and property of the ontology are constructed and similarity between the elements of the two models is computed with pairwise calculation of their identifying vectors' cosine distance. Essentially, the intuition is that the relational schema is also translated to a graph and some form of elementary graph matching is performed. An obvious restricting drawback is the choice of an element's name as its description, which immediately eliminates the possibility of matching e.g. a relation named *academic_staff* with a *faculty* class. MARSON then uses this simple correspondences between relations and OWL classes in order to test the consistency of the attributes and properties mappings discovered, while it exploits database data and ontology individuals to discover more complex mappings, e.g. finding the categorical attribute of a relation that splits it into subsets which in turn correspond to disjoint OWL subclasses of a superclass. This is performed by lexically comparing instances of the subclasses to database values. Then, the information gain for every categorical attribute with respect to the matching sets is computed. MARSON represents a clear effort to automatize the process of finding mappings as much as possible. However, no method to find the categorical attributes of a relation is described (thus, information gain for every attribute of a relation should be computed) and the computational efficiency of this approach is questionable, as the number of string similarity checks is directly related to the data volume of the database.

OntoMat-Reverse [72] is part of the OntoMat annotation framework, focusing on dynamic Web page annotation. OntoMat-Reverse automatically proposes mappings between the database schema, from which the data of the dynamic Web page is retrieved, and an already existing ontology. OntoMat-Reverse complements the reverse engineering rules of Stojanovic *et al.* [67] with a lexical similarity metric. In essence, it recognizes special structure designs in the database schema (in much the same way as the methods in Section 4.2.2) and tries to match them lexically with the classes and properties of the existing ontology. The efficiency of this tool is bounded by both the inherent inability of reverse engineering methods to interpret correctly all database design choices and the reliance on lexical methods to find the best match. Rationally, validation from an expert is required in the end of the process. The same matching approach is followed by **D2OMapper** [75], which uses the ER2WO [74] translation rules from an ER model to an OWL ontology. However, as ER2WO is based on an earlier proposal for mapping to DL ontologies [23] which mapped every binary relationship of the ER model to an ontology class, it is highly unlikely that an OWL ontology will use such design practices. Thus, the success of this tool in finding correct mappings for an ontology that is *not* produced by ER2WO is doubtful.

Overall, tools in this group range from manual mapping definition tools to automatic mapping discovery frameworks, with the latter based on simplifying assumptions, such as the lexical proximity in the naming of database and ontology elements, leading to the overestimation of their efficiency. The assumption that is also implied in all of the above tools is the fact that the already designed ontology is carefully chosen to closely match the subject domain of the database schema.

6. Overview of Methods

As already implied in Sections 4 and 5, the approaches and tools proposed for bringing relational databases into the Semantic Web are diverse and hence, difficult to compare. In this section, we attempt to concentrate the methods reviewed and present an overview of them.

Referring to the problem of creating a new ontology from a relational database, analyzed in Section 4, we list in Table 2 the sources of semantics for every proposed method. Sources of semantics considered include the database schema, database data, queries or data manipulation language (DML) statements, input from a human expert and thesauri or external vocabularies. We have omitted the tools of Section 4.1, because by definition they only consider database schema as input. We can see that all of the tools that do not perform schema analysis rely entirely on user input to define the mappings, unless they perform an automatic exportation to RDF according to the "table-to-class, column-to-predicate" approach. On the contrary, most tools that perform database schema analysis, depend solely on schema and sometimes, on user for validation of the created ontology. Exploitation of data and

Table 2
List of input information sources for methods of Section 4.2

	Input sources				
Approach	Schema	Data	Queries/DML	Other Sources	User
Astrova (2004) [7]	X	X			
Buccella *et al.* (2004) [21]	X				
D2RQ [19]	X[a]				X
DB2OWL [34]	X				
Kashyap (1999) [40]	X		X	Additional database schema, thesauri & controlled vocabularies	X
METAmorphoses [68]					X
R2O [66]	X				
RDBToOnto [24]	X	X			
Shen *et al.* (2006)[64]	X				
SOAM [48]	X			Lexical repositories	X
SQL2OWL [2]	X	X		ER model for validation	X
SquirrelRDF [61]					X
Stojanovic *et al.* (2002) [67]	X				
Tirmizi *et al.* (2008) [69]	X				
Triplify [9]					X
Virtuoso RDF Views [20]					X

[a] Schema analysis is only performed by D2RQ's *generate-mapping* script

other sources of semantics is rare and often not part of a formalized method.

Table 3 sums up the features of the most notable approaches, taking inspiration from other surveys that have been carried out for the same subject [42,60,63,76]. We have included as characterizing features the requirement of existence of an ontology, the degree of automation of each method, the data access method for the generated ontology, the language in which mappings are expressed, ontology language and whether the tool is available or not. Possible access methods include batch exportation, also known as *Extract-Transform-Load (ETL)*, access through *SPARQL queries* or *Linked Data* HTTP access. ETL approaches migrate the entire database (or, to be more precise, the part of the database instance that participates in the transformation) in a materialized ontology, access through SPARQL involves rewriting SPARQL queries to SQL queries that are executed by the relational database and results are transformed to SPARQL results form and Linked Data access implies that the URL generation mechanism is fully reversible, thus an HTTP request can be translated back to a database record. We should note that this feature applies only to tools that create a new ontology, while the ontology language feature refers either to the generated or the existing ontology, depending on whether an existing ontology is required for the procedure or not. The degree of automation for the tools of Section 4.2 can also be inferred from Table 2 by inspection of the *Schema* and *User* columns.

Inspecting Table 3, we can reach to some first conclusions, expected, to some point, from the analysis of Sections 4 and 5. To begin with, tools that create a new ontology describing the concepts of the database schema (e.g. Relational.OWL, RDB2ONT) are automatic and do not need to express in some form the mapping between the database schema and the ontology, since this is straightforward. Furthermore, tools that create a domain-specific ontology without performing database schema analysis (e.g. D2RQ, Virtuoso RDF Views) do not require an ontology for their task, but existing vocabularies can be reused in the case of manual mapping definition. This feature is not included in Table 3, since it applies to this group of tools only. We can also observe that some tools discovering mappings between a database instance and an existing ontology do not exploit these mappings in order to access data through SPARQL or translate it to ontology individuals.

We can also observe that the degree of automation of a tool is inversely correlated with the amount of

Table 3
Features of methods reviewed.

Approach	Existence of Ontology	Automation	Data Access	Mapping Language	Ontology Language	Tool Availability
Automapper (Asio SBRD) [33]	No	Automatic	SPARQL	Automapper Mapping Language (RDF)	OWL+SWRL	Commercial
D2OMapper [75]	Yes	Semi-Automatic	-	XML	OWL DL	No
D2RQ / D2R Server [19,17]	No	Automatic / Manual	ETL / SPARQL / Linked Data	D2RQ Mapping Language (RDF)	RDF	Yes
DartGrid [25]	Yes	Manual	SPARQL	RDF/XML	OWL DL	No
DataMaster [52]	No	Automatic	ETL	-	OWL DL/Full	Yes
DB2OWL [34]	No	Automatic	SPARQL	-	OWL DL	No
Dolbear, Hart (2008) [31]	No	Automatic	SPARQL	-	OWL DL	No
FDR2 [44]	No	Automatic	ETL	-	RDFS	No
Kupfer *et al.* (2006) [45]	No	Automatic	ETL	-	OWL-Lite	No
MAPONTO [4]	Yes	Semi-Automatic	-	-	OWL DL	Yes
MARSON [38]	Yes	Automatic	-	-	OWL DL	No
METAmorphoses [68]	No	Manual	ETL	METAmorphoses mapping Language (XML)	RDF	Yes
OntoMat-Reverse [72]	Yes	Semi-Automatic	ETL	F-Logic	F-Logic	No
R_2O / ODEMapster [11]	Yes	Manual	ETL / SPARQL	R_2O language	OWL	Yes
RDB2ONT [70]	No	Automatic	ETL	-	OWL DL	No
RDBToOnto [24]	No	Automatic	ETL	-	OWL	Yes
Relational.OWL [53]	No	Automatic	ETL / SPARQL	-	OWL Full	Yes
SOAM [48]	No	Semi-Automatic	ETL	-	OWL DL	No
SQL2OWL [2]	No	Semi-Automatic	ETL	-	OWL DL	No
SquirrelRDF [61]	No	Manual	ETL	SquirrelRDF Mapping Language (RDF)	RDF	Yes
Tether [22]	No	Semi-Automatic	ETL	-	RDFS	No
Triplify [9]	No	Manual	ETL / Linked Data	SQL	RDF	Yes
Ultrawrap [62]	No	Automatic	SPARQL	-	OWL DL	No
Virtuoso RDF Views [20]	No	Automatic / Manual	ETL / SPARQL / Linked Data	Virtuoso Meta-Schema Language	RDF	Yes
VisAVis [41]	Yes	Manual	RDQL	SQL	OWL DL	Yes

database semantics the tool elicits correctly, i.e. its semantic awareness. This happens mainly because human experts are the most reliable source of semantics of a database schema. Hence, purely manual tools, where mappings are given by the user, capture by definition the true meaning of the logical schema, whereas purely automatic tools can rarely achieve such levels of successful interpretation. This is depicted in Figure 2, where the categories of methods, as defined in this paper, are placed in a two-dimensional plane, where the horizontal axis refers to the degree of automation and the vertical axis to the level of semantic awareness. The placement of automatic creation of "database schema" and domain ontology problems to the lowest right part of the diagram, as well as the presence of manual approaches in the upper left part follows natu-

rally from the above. Approaches who create an ontology by performing analysis of the database schema are at large semi-automatic with the user simply validating the final result, thus this class is placed towards the middle of the plane. The same goes for semi-automatic mapping discovery approaches, with the only difference that some of them may achieve greater semantic awareness from the latter group of tools, since they often start with user input and base their more sophisticated techniques to this input rather than simply reverse engineering the database schema. User input is vital for these approaches, in contrast with schema-aware creation of ontology, where user validation is an optional step. Finally, automatic mapping discovery tools are less accurate than their semi-automatic counterparts, as they rely more on lexical distance metrics. The position of mapping discovery tools in the plane clearly shows the trade-off between the degree of automation and semantic awareness achieved. Unfortunately, tools that approach the upper right part of the diagram, i.e. automatic tools with high semantic awareness, do not seem to exist yet.

7. Discussion

In this paper, we have reviewed methods and tools that aim to bring relational databases into Semantic Web and render them accessible to a new exciting group of tools and applications. The problem of making relational databases compatible with Semantic Web technologies has been broadly referred to as database to ontology mapping. However, this term is somewhat vague, as it includes solutions to different problems. We have tried to bring some order to the large number of tools proposed in the literature by classifying them in clearly discernible groups. These groups comprise tools that create a new ontology mirroring a database instance, methods that a create a new domain ontology, either by analyzing the database schema to extract more semantics or by merely extracting database contents based on some user-defined mapping, and tools that allow the definition of correspondences between a database instance and an existing ontology.

Tools that create a new ontology having the database schema as domain, presented in Section 4.1 do not solve the problem of adding semantics to the database and therefore, do not significantly contribute towards the solution of problems outlined in Section 2. On the other hand, tools that extract RDF without analyzing the database schema for mining extra semantics (outlined in Section 4.2.1) are very popular and efficient, as they usually rely on some user-defined mapping expressed in some proprietary language and allow the reuse of existing vocabularies following Semantic Web best practices. These tools actually annotate database contents with as accurate semantics as possible (provided that the user is familiar with the domain), but do not provide any graphical interfaces for human domain experts with no programming skills.

Methods that attempt to analyze the database instance in order to create a new ontology are automatic or semi-automatic, in the latter case requiring the user to validate the resulting ontology. Table 2 reveals that the vast majority of methods in this category consider only database schema as source of semantics. Very few of them take into account data, queries on the database and other machine-analyzable sources and they often do not suggest a formal method but only give some intuitive examples. Essentially, such tools try to reverse engineer the logical schema of the database and, therefore, considering diverse knowledge sources for this task increases the possibility that the generated ontology will better capture the original conceptual model.

As far as methods trying to discover mappings between a database instance and a predefined ontology (analyzed in Section 5) are concerned, they range from manual graphical tools to automatic ones. The tools claiming to be automatic often rely on lexical proximity measures between database elements and ontology concepts, an assumption that may not hold in reality thus limiting successful discovery of matches. An important issue here is the reusability of the mappings produced by such tools. Table 3 reveals that every tool uses a different way to represent internally the mapping that is automatically discovered or defined by the user. Work on Relational to RDF Mapping Language (R2RML)[6] by the W3C RDB2RDF Working Group, will hopefully alleviate this problem and enable the sharing and reuse of mappings that will no longer be shut off in implementation-specific formats.

A somewhat surprising finding is the fact that, despite the wealth and variety of proposed approaches, less than half of the reviewed tools are available to the user and even less are still maintained. Thus, the large number of proposed approaches is not reflected on an equally large number of ready-to-use implementations. Equally impressive is the fact that none of the

[6] http://www.w3.org/2001/sw/rdb2rdf/r2rml/

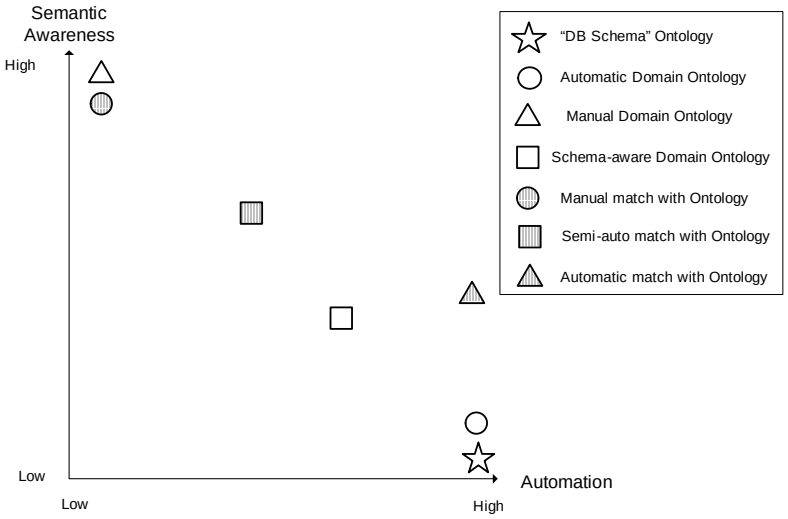

Fig. 2. Location of approaches in the semantic awareness vs. automation plane.

approaches considers OWL 2 profiles as the target ontology language, either for the ontology creation or the mapping discovery problem.

Evaluation and comparison of the reviewed tools is also a difficult issue, since there is no suitable benchmark that will measure the quality of the discovered mappings or the accuracy of the created ontology. The Berlin SPARQL benchmark [18] is not suitable for this purpose, as it can only be applied to tools that rewrite SPARQL queries to SQL queries, such as D2RQ and Virtuoso RDF Views, in order to compare their performance with native triple stores. [38] and [5] evaluate the mappings discovered by MARSON and MAPONTO respectively by using as reference the mappings devised by human experts and measuring the precision and recall for their tools. A benchmark suited for the mapping discovery problem would construct pairs of database schemas following various design practices and corresponding ontologies and measure the success of the tools using as reference a set of defined "correct" mappings. Similarly, for the ontology generation problem, the created ontology could be compared with a predefined exemplar ontology that should have been generated from the given database schema.

Another feature that current implementations completely lack is the dynamic adaptation of the mapping when the database schema or/and the ontology are modified. The most notable approach handling this issue is [6], which uses the theory underlying MAPONTO to propose appropriate modifications for the mappings and either the database schema or the ontology, depending on which is the one having changed.

Finally, SPARQL should not only be used for data retrieval from relational databases, but for data updates as well. The SPARQL recommendation is being constantly updated to include more features and the number of endpoints that implement the SPARQL 1.1 Update specification[7] and thus allow for RDF graph updates are increasing. However, the problem of updating relational data through SPARQL [37] is similar to the classic database view update problem and is certainly not a trivial one.

Acknowledgements

Dimitrios-Emmanuel Spanos wishes to acknowledge the financial support of 'Alexander S. Onassis Public Benefit Foundation', under its Scholarships Programme.

References

[1] S. Abiteboul, R. Hull and V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.

[2] N. Alalwan, H. Zedan and F. Siewe, Generating OWL Ontology for Database Integration, *2009 Third International Conference on Advances in Semantic Processing (SEMAPRO '09)*, pp. 22–31, IEEE, 2009.

[7] http://www.w3.org/TR/sparql11-update/

[3] R. Alhajj, Extracting the Extended Entity-Relationship Model from a Legacy Relational Database, *Information Systems*, **28**(6), pp. 597–618, 2003.

[4] Y. An, A. Borgida, R. J. Miller and J. Mylopoulos, A Semantic Approach to Discovering Schema Mapping Expressions, *IEEE 23rd International Conference on Data Engineering (ICDE'07)*, pp. 206–215, IEEE, 2007.

[5] Y. An, A. Borgida and J. Mylopoulos, Discovering the Semantics of Relational Tables through Mappings, *Journal on Data Semantics*, **7**, pp. 1–32, 2006.

[6] Y. An, X. Hu and I.-y. Song, Round-Trip Engineering for Maintaining Conceptual-Relational Mappings, *Proceedings of 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), LNCS 5074*, pp. 296–311, Springer, 2008.

[7] I. Astrova, Reverse Engineering of Relational Databases to Ontologies, *Proceedings of the First European Semantic Web Symposium (ESWS 2004), LNCS 3053*, pp. 327–341, 2004.

[8] I. Astrova, N. Korda and A. Kalja, Rule-Based Transformation of SQL Relational Databases to OWL Ontologies, *Proceedings of the 2nd International Conference on Metadata & Semantics Research*, 2007.

[9] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann and D. Aumueller, Triplify: Light-Weight Linked Data Publication from Relational Databases, *Proceedings of the 18th International Conference on World Wide Web*, pp. 621–630, ACM, 2009.

[10] F. Baader, D. L. McGuinness, P. F. Patel-Schneider and D. Nardi, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003.

[11] J. Barrasa - Rodriguez and A. Gómez-Pérez, Upgrading Relational Legacy Data to the Semantic Web, *Proceedings of the 15th International Conference on World Wide Web*, pp. 1069–1070, ACM New York, NY, USA, 2006.

[12] J. Barrasa, O. Corcho and A. Gómez-Pérez, R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language, *Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB 2004)*, Springer, 2004.

[13] D. Beckett and J. Grant, SWAD-Europe Deliverable 10.2: Mapping Semantic Web Data with RDBMSes, 2003, URL `http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/`.

[14] A. Behm, A. Geppert and K. R. Dittrich, On The Migration of Relational Schemas and Data to Object-Oriented Database Systems, *Proceedings of the 5th Conference on Re-Technologies for Information Systems*, pp. 13–33, 1997.

[15] C. Ben Necib and J.-C. Freytag, Semantic Query Transformation Using Ontologies, *9th International Database Engineering & Application Symposium (IDEAS'05)*, pp. 187–199, IEEE, 2005.

[16] T. Berners-Lee, Relational Databases on the Semantic Web, 1998, URL `http://www.w3.org/DesignIssues/RDB-RDF.html`.

[17] C. Bizer and R. Cyganiak, D2R Server - Publishing Relational Databases on the Semantic Web, *Proceedings of the 5th International Semantic Web Conference*, 2006.

[18] C. Bizer and A. Schultz, The Berlin SPARQL Benchmark, *International Journal On Semantic Web and Information Systems*, **5**(2), pp. 1–24, 2009.

[19] C. Bizer and A. Seaborne, D2RQ - Treating non-RDF Databases as Virtual RDF Graphs, *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

[20] C. Blakeley, Virtuoso RDF Views Getting Started Guide, 2007.

[21] A. Buccella, M. R. Penabad, F. R. Rodriguez, A. Farina and A. Cechich, From Relational Databases to Ontologies, *Proceedings of 6th Russian Conference on Digital Libraries*, 2004.

[22] K. Byrne, Having Triplets - Holding Cultural Data as RDF, *Proceedings of the ECDL 2008 Workshop on Information Access to Cultural Heritage*, 2008.

[23] D. Calvanese, M. Lenzerini and D. Nardi, Unifying Class-Based Representation Formalisms, *J. of Artificial Intelligence Research*, **11**, pp. 199–240, 1999.

[24] F. Cerbah, Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies, *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT '08)*, pp. 553–557, IEEE, 2008.

[25] H. Chen, Z. Wu, H. Wang and Y. Mao, RDF/RDFS-Based Relational Database Integration, *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*, pp. 94–103, IEEE, 2006.

[26] P. P.-S. Chen, The Entity-Relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems (TODS)*, **1**(1), pp. 9–36, 1976.

[27] R. H. Chiang, T. M. Barron and V. C. Storey, Reverse Engineering of Relational Databases: Extraction of an EER Model from a Relational Database, *Data & Knowledge Engineering*, **12**(2), pp. 107–142, 1994.

[28] E. Codd, A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, **13**(6), p. 387, 1970.

[29] C. Curino, G. Orsi, E. Panigati and L. Tanca, Accessing and Documenting Relational Databases through OWL Ontologies, *Proceedings of the 8th International Conference on Flexible Query Answering Systems, LNCS 5822*, pp. 431–442, Springer, 2009.

[30] S. Das and J. Srinivasan, Database Technologies for RDF, *5th International Summer School 2009 on Reasoning Web. Semantic Technologies for Information Systems, LNCS 5689*, pp. 205–221, 2009.

[31] C. Dolbear and G. Hart, Ontological Bridge Building - Using Ontologies to Merge Spatial Datasets, *Proceedings of the AAAI Spring Symposium on Semantic Scientific Knowledge Integration (AAAI/SSS Workshop)*, 2008.

[32] M. Fahad, ER2OWL: Generating OWL Ontology from ER Diagram, *Proceedings of 5th IFIP International Conference on Intelligent Information Processing*, pp. 28–37, Springer, 2008.

[33] M. Fisher, M. Dean and G. Joiner, Use of OWL and SWRL for Semantic Relational Database Translation, *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, 2008.

[34] R. Ghawi and N. Cullot, Database-to-Ontology Mapping Generation for Semantic Interoperability, *3rd International Workshop on Database Interoperability (InterDB 2007), held in conjunction with VLDB 2007*, 2007.

[35] A. Gomez-Perez, O. Corcho-Garcia and M. Fernandez-Lopez, *Ontological Engineering*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[36] J. Hendler, Web 3.0: Chicken Farms on the Semantic Web, *IEEE Computer*, **41**(1), pp. 106–108, 2008.

[37] M. Hert, G. Reif and H. C. Gall, Updating Relational Data via SPARQL/Update, *Proceedings of the 2010 EDBT/ICDT Work-*

shops (EDBT '10), 2010.

[38] W. Hu and Y. Qu, Discovering Simple Mappings between Relational Database Schemas and Ontologies, *Proceedings of the 6th International Semantic Web and 2nd Asian Semantic Web Conference, LNCS 4825*, p. 225, Springer, 2007.

[39] P. Johannesson, A Method for Transforming Relational Schemas into Conceptual Schemas, *10th International Conference on Data Engineering*, pp. 190–201, IEEE, 1994.

[40] V. Kashyap, Design and Creation of Ontologies for Environmental Information Retrieval, *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management*, 1999.

[41] N. Konstantinou, D.-E. Spanos, M. Chalas, E. Solidakis and N. Mitrou, VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents, *Proceedings of the CAiSE'06 3rd International Workshop on Web Information Systems Modeling (WISM'06)*, pp. 1050–1061, 2006.

[42] N. Konstantinou, D.-E. Spanos and N. Mitrou, Ontology and Database Mapping: A Survey of Current Implementations and Future Directions, *Journal of Web Engineering*, **7**(1), pp. 1–24, 2008.

[43] N. Konstantinou, D.-E. Spanos, P. Stavrou and N. Mitrou, Technically Approaching the Semantic Web Bottleneck, *International Journal of Web Engineering and Technology*, **6**(1), pp. 83–111, 2010.

[44] M. Korotkiy and J. L. Top, From Relational Data to RDFS Models, *Proceedings of the 4th International Conference on Web Engineering (ICWE 2004), LNCS 3140*, pp. 430–434, Springer, 2004.

[45] A. Kupfer, S. Eckstein, K. Neumann and B. Mathiak, Handling Changes of Database Schemas and Corresponding Ontologies, *Proceedings of the ER 2006 Workshops: Advances in Conceptual Modeling - Theory and Practice, LNCS 4231*, pp. 227–236, Springer, 2006.

[46] N. Lammari, I. Comyn-Wattiau and J. Akoka, Extracting Generalization Hierarchies from Relational Databases: A Reverse Engineering Approach, *Data & Knowledge Engineering*, **63**(2), pp. 568–589, 2007.

[47] M. Lenzerini, Data Integration: A Theoretical Perspective, *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 233–246, 2002.

[48] M. Li, X. Du and S. Wang, A Semi-Automatic Ontology Acquisition Method for the Semantic Web, *Proceedings of the 6th International Conference on Advances in Web-Age Information Management (WAIM 2005), LNCS 3739*, pp. 209–220, Springer, 2005.

[49] L. Lubyte and S. Tessaris, Automatic Extraction of Ontologies Wrapping Relational Data Sources, *Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA 2009), LNCS 5690*, pp. 128–142, Springer, 2009.

[50] A. Maedche and S. Staab, Ontology Learning for the Semantic Web, *IEEE Intelligent Systems*, **16**(2), pp. 72–79, 2001.

[51] B. Motik, I. Horrocks and U. Sattler, Bridging the Gap between OWL and Relational Databases, *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, pp. 807–816, ACM Press, New York, New York, USA, 2007.

[52] C. Nyulas, M. O'Connor and S. Tu, DataMaster - a Plug-in for Importing Schemas and Data from Relational Databases into Protégé, *10th International Protégé Conference*, 2007.

[53] C. Pérez De Laborda and S. Conrad, Relational.OWL - A Data and Schema Representation Format Based on OWL, *Proceedings of the Second Asia-Pacific Conference on Conceptual Modeling (APCCM2005)*, pp. 89–96, 2005.

[54] C. Pérez De Laborda and S. Conrad, Database to Semantic Web Mapping using RDF Query Languages, *25th International Conference on Conceptual Modeling (ER 2006), LNCS 4215*, pp. 241–254, 2006.

[55] C. Pérez De Laborda, M. Zloch and S. Conrad, RDQuery - Querying Relational Databases on-the-fly with RDF-QL, *Posters and Demos of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*, 2006.

[56] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini and R. Rosati, Linking Data to Ontologies, *Journal on Data Semantics*, **10**, pp. 133–173, 2008.

[57] W. J. Premerlani and M. R. Blaha, An Approach for Reverse Engineering of Relational Databases, *Communications of the ACM*, **37**(5), pp. 42–49, 1994.

[58] R. Ramakrishnan and J. Gehrke, *Database Management Systems*, McGraw-Hill, second ed., 2002.

[59] S. Ramanathan and J. Hodges, Extraction of Object-Oriented Structures from Existing Relational Databases, *ACM SIGMOD Record*, **26**(1), p. 64, 1997.

[60] S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau, S. Auer, J. Sequeda and A. Ezzat, A Survey of Current Approaches for Mapping of Relational Databases to RDF, 2009.

[61] A. Seaborne, D. Steer and S. Williams, SQL-RDF, *W3C Workshop on RDF Access to Relational Databases*, 2007.

[62] J. F. Sequeda, R. Depena and D. P. Miranker, Ultrawrap: Using SQL Views for RDB2RDF, *8th International Semantic Web Conference (ISWC2009)*, 2009.

[63] J. F. Sequeda, S. H. Tirmizi, O. Corcho and D. P. Miranker, Direct Mapping SQL Datababases to the Semantic Web: A Survey (Technical Report TR-09-04), 2009.

[64] G. Shen, Z. Huang, X. Zhu and X. Zhao, Research on the Rules of Mapping from Relational Model to OWL, *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED 2006)*, vol. 216, 2006.

[65] E. Sirin and J. Tao, Towards Integrity Constraints in OWL, *Proceedings of the 2009 International Workshop on OWL: Experiences and Directions (OWLED 2009)*, 2009.

[66] K. Sonia and S. Khan, R2O Transformation System: Relation to Ontology Transformation for Scalable Data Integration, *Proceedings of the 2008 International Symposium on Database Engineering & Applications*, pp. 291–295, ACM New York, NY, USA, 2008.

[67] L. Stojanovic, N. Stojanovic and R. Volz, Migrating Data-Intensive Web Sites into the Semantic Web, *Proceedings of the 2002 ACM Symposium on Applied computing*, pp. 1100–1107, ACM New York, NY, USA, 2002.

[68] M. Svihla and I. Jelinek, Two Layer Mapping from Database to RDF, *Proceedings of Electronic Computers and Informatics (ECI)*, 2004.

[69] S. H. Tirmizi, J. F. Sequeda and D. P. Miranker, Translating SQL Applications to the Semantic Web, *Proceedings of 19th International Conference on Database and Expert Systems Applications (DEXA 2008), LNCS 5181*, pp. 450–464, Springer, 2008.

[70] Q. Trinh, K. Barker and R. Alhajj, RDB2ONT: A Tool for Gen-

erating OWL Ontologies From Relational Database Systems, *Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pp. 170–178, IEEE, 2006.

[71] S. R. Upadhyaya and P. S. Kumar, ERONTO: a Tool for Extracting Ontologies from Extended E/R Diagrams, *Proceedings of the 2005 ACM Symposium on Applied Computing*, pp. 666 – 670, ACM, 2005.

[72] R. Volz, S. Handschuh, S. Staab, L. Stojanovic and N. Stojanovic, Unveiling the Hidden Bride: Deep Annotation for Mapping and Migrating Legacy Data to the Semantic Web, *Web Semantics: Science, Services and Agents on the World Wide Web*, **1**(2), pp. 187–206, 2004.

[73] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner, Ontology-Based Integration of Information - A Survey of Existing Approaches, *IJCAI-01 Workshop: Ontologies and Information Sharing*, pp. 108–117, 2001.

[74] Z. Xu, X. Cao, Y. Dong and W. Su, Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies, *Proceedings of Advances in Knowledge Discovery and Data Mining 8th Pacific-Asia Conference (PAKDD 2004), LNCS 3056*, pp. 464–475, Springer-Verlag New York Inc, 2004.

[75] Z. Xu, S. Zhang and Y. Dong, Mapping between Relational Database Schema and OWL Ontology for Deep Annotation, *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings) (WI'06)*, pp. 548–552, IEEE, 2006.

[76] S. Zhao and E. Chang, From Database to Semantic Web Ontology: An Overview, *On the Move to Meaningful Internet Systems: OTM 2007 Workshops, LNCS 4806*, pp. 1205–1214, Springer, 2007.