Semantic Web 0 (0) 1–41
IOS Press

Bringing Relational Databases into the Semantic Web: A Survey

Editor(s): Jie Tang, Tsinghua University, China
Solicited review(s): Zi Yang, Carnegie Mellon University, USA; Yuan An, Drexel University, USA; Ling Chen, University of Technology, Sydney, Australia; Juanzi Li, Tsinghua University, China

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 various 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, benefits, challenges and solutions for each one 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 and more, the Semantic Web (SW) has grown from an abstract futuristic vision, mainly existing in the head of its inspirer, Tim Berners-Lee, into an ever approaching reality of a global web of interlinked data with well-defined meaning. Standard languages and technologies have been proposed and are constantly evolving in order to serve as the building blocks for this "next generation" Web, relevant tools are being developed and gradually reaching maturity, while numerous real world applications already give an early taste of the benefits the Semantic Web is about to bring in various domains, as diverse as life sciences, environmental monitoring, cultural heritage, e-Government and business process management. This evident progress is the result of years-long research and it comes as no surprise that, nowadays, Semantic Web is perceived as a multidisciplinary research field on its own, combining and gaining expertise from other scientific fields, such as artificial intelligence, information science, algorithm and complexity theory, database theory and computer networks, to name a few.

The participation of databases and their role in this evolving Web setting has been investigated from the very beginning of the Semantic Web conception, not only because it was initially compared to "a global database" [23], but also because this – new at the time – research field could take advantage of the great experience and maturity of the database field. However, the collaboration and exchange of ideas between

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

these two fields was not unidirectional: the database community quickly recognized the opportunities arising from a close cooperation with the Semantic Web field [113] and how the latter could offer solutions to long-standing issues and provide inspiration to several database subcommunities, interested in heterogeneous database integration and interoperability, distributed architectures, deductive databases, conceptual modelling and so on.

Attempts to combine these two different worlds originally focused on the reconciliation of the discrepancies among the two most representative and dominant technologies of each world: relational databases and ontologies. This problem is also known as the *database to ontology mapping problem*, which is subsumed by the broader *object-relational impedance mismatch problem* and is due to the structural differences among relational and object-oriented models. Correspondences between the relational model and the RDF graph model, which is a key component of the Semantic Web, were also investigated and a W3C Working Group[1] has been formed to examine this issue and propose related standards. Nevertheless, the definition of a theoretically sound mapping or transformation between the mentioned models is not an end on its own. The motivation driving the consideration of mappings among relational databases and Semantic Web technologies is multifold, leading to separate problems, where mappings are discovered, defined and used in a different way for each problem case.

Originally, database systems were considered by the Semantic Web community as an excellent means for efficient ontology storage [19], because of their known and well evidenced performance benefits. This consideration has led to the development and production of several database systems, especially optimized for the persistent storage, maintenance and querying of SW data [47]. Such systems are informally known as *triple stores*, since they are specifically tailored for the storage of RDF statements, which are also referred to as triples. This sort of collaboration between database and Semantic Web specifies a data and information flow from the latter to the former, in the form of population of specialized databases, that often have some predefined structure, with SW data.

A different, perhaps more interesting research line takes as starting point an existing and fully functional relational database and seeks ways to extract information and render it suitable for use from a Semantic Web perspective. In this case, motivation is shifted from the efficient storage and querying of existing ontological structures to problems, such as database integration, ontology learning, mass generation of SW data, ontology-based data access and semantic annotation of dynamic Web pages. These problems have been investigated in the relevant literature, each one touching on a different aspect of the database to ontology mapping problem. Unfortunately, this term has been freely used to describe most of the aforementioned issues, creating slight confusion regarding the goal and the challenges faced for each one of them. Hence, in this paper, we take a look at approaches that do one or more of the following:

- create from scratch a new ontology based on information extracted from a relational database,
- generate RDF statements that conform to one or more predefined ontologies and reflect the contents of a relational database,
- answer semantic queries directly against a relational database, and
- discover correspondences between a relational database and a given ontology

We attempt to define and distinguish between these closely related but distinct problems, analyze methods and techniques employed in order to deal with them, identify features of proposed approaches, classify them accordingly and present a future outlook for this much researched issue.

The paper is structured as follows: Section 2 mentions the motivations and benefits of the database to ontology mapping problem, while Section 3 gives a short background and defines some terminology of database systems and Semantic Web technologies. Section 4 presents a total classification of relational database to ontology mapping approaches and lists descriptive parameters used for the description of each approach. The structure of the rest of paper is largely based on this taxonomy. Section 5 deals with the problem of creating a new ontology from a relational database and is further divided in subsections that distinguish among the creation of a database schema ontology (Section 5.1) and a domain-specific ontology (Section 5.2) as well as among approaches that rely extensively on the analysis of the database schema (Section 5.2.2) and approaches that do not (Section 5.2.1). Section 6 investigates the problem of discovering and defining mappings between a relational database and

[1] RDB2RDF Working Group: `http://www.w3.org/2001/sw/rdb2rdf/`

one or more existing ontologies. Section 7 sums up the main points of the paper, giving emphasis on the challenges faced by each category of approaches, while Section 8 gives some insight on future directions and requirements for database to ontology mapping solutions.

2. Motivation and Benefits

The significance of databases from a Semantic Web perspective is evident from the multiple benefits and use cases a database to ontology mapping can be used in. After all, the problem of mapping a database into Semantic Web did not emerge as a mere exercise of transition from one representation model to another. It is important to identify the different motivations and problems implicating interactions between relational databases and SW technologies, in order to succeed a clear separation of goals and challenges. That is not to say that methods and approaches presented here correspond strictly to one particular use case, as there are quite a few versatile tools that kill two (or more) birds with one stone. In the following, we present some of the benefits that can be achieved from the interconnection of databases and ontologies.

Semantic annotation of dynamic web pages. An aspect of the Semantic Web vision is the transformation of the current Web of documents to a Web of data. A straightforward way to achieve this would be to annotate HTML pages, which specify the way their content is presented and are only suitable for human consumption. HTML pages can be semantically annotated with terms from ontologies, making their content suitable for processing by software agents and web services. Such annotations have been facilitated considerably since the proposal of the RDFa recommendation[2] that embeds in XHTML tags references to ontology terms. However, this scenario does not work quite well for dynamic web pages that retrieve their content directly from underlying databases: this is the case for content management systems (CMS), fora, wikis and other Web 2.0 sites [14]. Dynamic web pages represent the biggest part of the World Wide Web, forming the so called Deep Web [54], which is not accessible to search engines and software agents, since these pages are generated in response to a web service or a web form interface request. It has been argued that, due to the infeasibility of manual annotation of every single dynamic page, a possible solution would be to "annotate" directly the underlying database schema, insofar as the web page owner is willing to reveal the structure of his database. This "annotation" is simply a set of correspondences between the elements of the database schema and an already existing ontology that fits the domain of the dynamic page content [125]. Once such mappings are defined, it would be fairly trivial to generate dynamic semantically annotated pages with the embedded content annotations derived in an automatic fashion.

Heterogeneous database integration. The resolution of heterogeneity is one of the most popular, long-standing issues in the database research field, that remains, to a large degree, unsolved. Heterogeneity occurs between two or more database systems when they use different software or hardware infrastructure, follow different syntactic conventions and representation models, or when they interpret differently the same or similar data [114]. Resolution of the above forms of heterogeneity allows multiple databases to be integrated and their contents to be uniformly queried. In typical database integration architectures, one or more conceptual models are used to describe the contents of every source database, queries are posed against a global conceptual schema and, for each source database, a wrapper is responsible to reformulate the query and retrieve the appropriate data. Ontology-based integration employs ontologies in lieu of conceptual schemas and therefore, correspondences between source databases and one or more ontologies have to be defined [126]. Such correspondences consist of mapping formulas that express the terms of a source database as a conjunctive query against the ontology (Local as view or LAV mapping), express ontology terms as a conjunctive query against the source database (Global as view or GAV mapping), or state an equivalence of two queries against both the source database and the ontology (Global Local as view or GLAV mapping) [80]. The type of mappings used in an integration architecture influences both the complexity of query processing and the extensibility of the entire system (e.g. in the case of GAV mappings, query processing is trivial, but the addition of a new source database requires redefinition of all the mappings; the inverse holds for LAV mappings). Thus, the discovery and representation of mappings between relational database schemas and ontologies constitute an integral

[2] RDFa Primer: http://www.w3.org/TR/xhtml-rdfa-primer/

part of a heterogeneous database integration scenario [8].

Ontology-based data access. Much like in a database integration architecture, ontology-based data access (OBDA) assumes that an ontology is linked to a source database, thus acting as an intermediate layer between the user and the stored data. The objective of an OBDA system is to offer high-level services to the end user of an information system who does not need to be aware of the obscure storage details of the underlying data source [98]. The ontology provides an abstraction of the database contents, allowing users to formulate queries in terms of a high-level description of a domain of interest. In some way, an OBDA engine resembles a wrapper in an information integration scenario in that it hides the data source-specific details from the upper levels by transforming queries against a conceptual schema to queries against the local data source. This query rewriting is performed by the OBDA engine, taking into account mappings between a database and a relevant ontology describing the domain of interest. The main advantage of an OBDA architecture is the fact that semantic queries are posed directly against a database, without the need to replicate its entire contents in RDF.

Apart from OBDA applications, a database to ontology mapping can be useful for *semantic rewriting of SQL queries*, where the output is a reformulated SQL query better capturing the intention of the user [21]. This rewriting is performed by substitution of terms used in the original SQL query with synonyms and related terms from the ontology. Another notable related application is the ability to query relational data using as context external ontologies [45]. This feature has been implemented in some database management systems[3], allowing SQL queries to contain conditions expressed in terms of an ontology.

Mass generation of Semantic Web data. It has been argued that one of the reasons delaying the Semantic Web realization is the lack of successful tools and applications showcasing the advantages of SW technologies [73]. The success of such tools, though, is directly correlated to the availability of a sufficiently large quantity of SW data, leading to a "chicken-and-egg problem" [62], where cause and effect form a vicious circle. Since relational databases are one of the most popular storage media holding the majority of data on the World Wide Web, a solution for the generation of a critical mass of SW data would be the, preferably automatic, extraction of relational databases' contents in RDF. This would create a significant pool of SW data, that would alleviate the inhibitions of software developers and tool manufacturers and, in turn, an increased production of SW applications would be anticipated. The term database to ontology mapping has been used in the literature to describe such transformations as well.

Ontology learning. The process of manually developing from scratch an ontology is difficult, time-consuming and error-prone. Several semi-automatic ontology learning methods have been proposed, extracting knowledge from free and semi-structured text documents, vocabularies and thesauri, domain experts and other sources [56]. Relational databases are structured information sources and, in case their schema has been modelled following standard practices [51] (i.e. based on the design of a conceptual model, such as UML or the Extended Entity Relationship Model), they constitute significant and reliable sources of domain knowledge. This is true especially for business environments, where enterprise databases are frequently maintained and contain timely data [129]. Therefore, rich ontologies can be extracted from relational databases by gathering information from their schemas, contents, queries and stored procedures, as long as a domain expert supervises the learning process and enriches the final outcome. Ontology learning is a common motivation driving database to ontology mapping when there is not an existing ontology for a particular domain of interest, a situation that frequently arose not so many years ago. Nevertheless, as years pass by, ontology learning techniques are mainly used to create a wrapping ontology for a source relational database in an ontology-based data access [110] or database integration [29] context.

Definition of the intended meaning of a relational schema. As already mentioned, standard database design practices begin with the design of a conceptual model, which is then transformed, in a step known as logical design, to the desired relational model. However, the initial conceptual model is often not kept alongside the implemented relational database schema and subsequent changes to the latter are not propagated back to the former, while most of the times these changes are not even documented at all. Usually, this results in databases that have lost the original intention of their designer and are very hard to be extended or

[3]Oracle and OpenLink Virtuoso are the most representative examples.

re-engineered to another logical model (e.g. an object-oriented one). Establishing correspondences between a relational database and an ontology grounds the original meaning of the former in terms of an expressive conceptual model, which is crucial not only for database maintenance but also for the integration with other data sources [38], and for the discovery of mappings between two or more database schemas [7,49]. In the latter case, the mappings between the database and the ontology are used as an intermediate step and a reference point for the construction of inter-database schema mappings.

Integration of database content with other data sources. Transforming relational databases into a universal description model, as RDF aspires to be, enables seamless integration of their contents with information already represented in RDF. This information can originate from both structured and unstructured data sources that have exported their contents in RDF, thus overcoming possible syntactic disparities among them. The Linked Data paradigm [60], which encourages RDF publishers to reuse popular vocabularies (i.e. ontologies), to define links between their dataset and other published datasets and reuse identifiers that describe the same real-world entity, further facilitates global data source integration, regardless of the data source nature. Given the uptake of the Linked Data movement during the last few years, which has resulted in the publication of voluminous RDF content (in the order of billion statements) from several domains of interest, the anticipated benefits of the integration of this content with data currently residing in relational databases as well as the number of potential applications harnessing it are endless.

3. Preliminaries

In this section, we make a short introduction on some aspects of database theory and Semantic Web technologies and define concepts that will be used throughout the paper. We first focus on conceptual and logical database design by visiting the Entity-Relationship and relational models and then, we briefly touch upon the RDF graph model and OWL ontology language.

3.1. Database theory fundamentals

The standard database design process mainly consists of three steps: a) conceptual, b) logical and c) physical database design [102]. Conceptual design usually follows after a preliminary phase where data and functional requirements for potential database users are analyzed. The output of the conceptual design is a *conceptual schema*, which describes from a high-level perspective the data that will be stored in the database and the constraints holding over it. The conceptual schema is expressed in terms of a conceptual model, the most widely used for this purpose being the **Entity-Relationship (ER) model**, proposed by Peter Chen [39]. The main features of the ER model are:

- *entity sets*, which are collections of similar objects, called *entities*.
- *attributes* associated with each entity set. An attribute can be either single-valued or multi-valued. All the entities belonging to a given entity set are described by the same attributes. For every entity set, a *key* is specified as a minimal set of attributes, the values of which uniquely identify an entity.
- *relationship sets*, which are sets of *relationship instances*, connecting n entities with each other. n is called the *degree* of the relationship set. A relationship set can have descriptive attributes, just like an entity set.
- *cardinality constraints*, which specify the minimum and maximum number of entities from a given entity set that can participate in a relationship instance from a given relationship set.
- *weak entity sets*, which consist of *weak entities*, the existence of which depends on the presence of another entity, called the *owner entity*. The relationship that connects a weak entity with its owner entity is called *identifying relationship*.

Since its original definition, the ER model has been further extended to include more characteristics that enhanced its expressiveness. The extended version of the ER model is known as the Extended Entity-Relationship (EER) model, which allows the definition of subclass/superclass relationships among entity sets and introduces, among others, the union type and aggregation features. A formal definition of the ER model can be found in [33].

Logical database design consists of building a *logical schema*, which conforms to a representational model that conceptually sits between the ER model and the physical model describing the low-level storage details of the database. Such models include the network, hierarchical and relational models, the most

popular one being the latter, proposed by Edgar Codd [41]. The main elements of the **relational model** are:

- *relations*, which intuitively can be thought of as tables.
- *attributes*, which can be regarded as columns of a table. Every attribute belongs to a given relation. A minimal set of attributes that uniquely define an element of a relation (i.e. a *tuple* or informally, a row of the table) is a *candidate key* for that relation. In case there exist more than one candidate keys, one of them is specified as the *primary key* of the relation and is used for the identification of this relation's tuples. A *foreign key* for a given relation, called parent relation, is a set of attributes that reference a tuple in another relation, called child relation.
- *domains*, which are sets of atomic constant values. Every attribute is related to a specific domain and all of its values are members of this domain.

Again, a formal definition of the relational model can be found in [3]. A **relation schema** is simply the name of a relation, combined with the names and domains of its attributes and a set of primary key and foreign key constraints. A *relation instance* is a set of tuples that have the same attributes as the corresponding relation schema and satisfy the relation schema constraints. A **relational database schema** is the set of relation schemas for all relations in a database, while a *relational database instance* is a set of relation instances, one for every relation schema in the database schema.

The transition from a conceptual schema expressed in terms of the EER model to a relational schema is performed according to well established standard algorithms [51], that define correspondences between constructs of the two models. A very rough sketch of these correspondences is presented in Table 1, which, although incomplete, clearly reveals that the transformation of the EER model to the relational one is not reversible. This is the source of the *database reverse engineering problem* i.e. the problem of recovering the initial conceptual schema from the logical one.

The relational schema obtained from a direct translation of the conceptual schema is not always optimal for data storage and may cause problems during data update operations. Thus, the relational schema is further refined in an additional *normalization* step, which minimizes the redundancy of stored information. In short, the normalization process decomposes relations of a relational database schema, in order to eliminate dependencies among relation attributes. The most prominent types of dependencies met in relational schemas are functional, inclusion, multi-valued and join dependencies. Depending on the type of dependencies eliminated, the decomposed relations are said to be in a specific *normal form*. Normal forms based on functional dependencies are the first, the second, the third and the Boyce-Codd normal form. Fourth and fifth normal form are based on multi-valued and join dependencies respectively[4]. Without going into further detail, it suffices to say that normalization, as an additional design step, incurs further complications to the database reverse engineering problem.

Much of the success of relational databases is due to SQL, which stands for Structured Query Language [1] and is the standard query language for relational databases. Despite the fact that SQL does not strictly follow Codd's relational model[5], it is relationally complete, i.e. all of the relational algebra operations (e.g. selection, projection, join, union) can be expressed as an SQL query. SQL serves not only as a declarative query language – the subset of SQL called Data Manipulation Language (DML) – but also as a database schema definition language – the subset of SQL called Data Definition Language (DDL). Thus, a relational database schema can often be expressed as an SQL DDL statement.

3.2. Semantic Web technologies

Moving on to the Semantic Web field, we begin with RDF, probably one of the most significant SW building blocks. The **RDF model** is a graph-based model based on the concepts of *resources* and *triples* or *statements*. A resource can be anything that can be talked about: a web document, a real-world entity or an abstract concept and is identified by a URI (Uniform Resource Identifier). 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 that form a directed labelled graph. Subjects and objects of RDF triples are represented by nodes, while predicates

[4]For more information on dependency types and normalization, see standard database theory textbooks [3,51,102].

[5]The SQL standard uses the terms table, column and row, instead of the equivalent relational model terminology of relation, attribute and tuple. It also allows the presence of identical rows in the same table, a case frequently observed in derived tables that hold the results of an SQL query.

Table 1
EER to relational model transformation

EER model		Relational model
Entity set		Relation
Attribute	Single-valued	Attribute
	Multi-valued	Relation and foreign key
Binary relationship set	1:1 or 1:N	Foreign key or relation
	M:N	Relation
n-degree relationship set ($n > 2$)		Relation and n foreign keys
Hierarchy of entity sets		One relation per entity set and foreign keys or one relation for all entity sets

are arcs pointing from the subject to the object of the triple. A special case of nodes are blank nodes, which are not identified by a URI. An RDF graph can be serialized in various formats, some of which are suitable mainly for machine consumption (such as the XML-based RDF/XML) and some are more easily understandable by humans (such as the plain text Turtle and N3 syntaxes[6]).

Another fundamental concept not only for Semantic Web, but information science in general, is the concept of **ontology**. In the context of computer science, several definitions have been given for ontology, from Tom Gruber's "ontology is an explicit specification of a conceptualization"[58] and Nicola Guarino's "ontology is a logical theory accounting for the intended meaning of a formal vocabulary"[59] to more concrete formal ones [46,69]. The abundance of ontology definitions stems from the broadness of the term, which includes ontologies of several logic formalisms and representation languages. Therefore, we slightly adapt the definition of Maedche and Staab [86] which is general enough to describe most ontologies. According to [86], an ontology is a set of primitives containing:

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

Ontologies can be classified according to the subject of the conceptualization: hence, among others, we can distinguish between knowledge representation, upper-level, domain and application ontologies [56]. Domain (or *domain-specific*) ontologies are the most common ones and provide a vocabulary for concepts of a specific domain of interest and their relationships. We will see in Section 5 that there are approaches that map relational databases to domain ontologies, where the domain can be either the relational model itself or the topic on which the contents of the database touch.

Several ontology languages have been proposed for the implementation of ontologies, but RDF Schema (RDFS) and Web Ontology Language (OWL) are the most prominent ones, especially in the Semantic Web domain. RDFS is a minimal ontology language built on top of RDF that allows the definition of:

- *classes* of individual resources.
- *properties*, connecting two resources.
- hierarchies of classes.
- hierarchies of properties.
- domain and range constraints on properties.

OWL, on the other hand, provides ontology designers with much more complex constructs than RDFS does and its popularity is constantly rising. As in RDFS, 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 is based on Description Logics (DL) [15], a family of languages that are subsets of first-order logic, and offers several dialects of differing expressiveness. Following the DL paradigm, statements in OWL can

[6]The most popular among the two, Turtle, has been put on W3C Recommendation track (Working Draft: `http://www.w3.org/TR/turtle/`).

be divided in two groups: the *TBox*, containing intensional knowledge (i.e. axioms about classes and properties) and the *ABox*, containing extensional knowledge (i.e. statements involving individuals). Daring an analogy with database systems, TBox could be compared to the database schema and ABox to the database instance. The combination of TBox and ABox, together with a provider of *reasoning* services, is often also called a *knowledge base*.

The first generation of the OWL language family, also known as OWL 1, defined three OWL species: OWL Lite, OWL DL and OWL Full, in order of increasing expressiveness. However, these species turned out not to be very practical for real-world applications. Therefore, the second generation of OWL language, OWL 2, introduced three profiles of different expressiveness, with overlapping feature sets: OWL 2 EL, OWL 2 RL and OWL 2 QL, each one best suited for a different application scenario[7]. Furthermore, OWL 2 DL and Full were defined as syntactic extensions of OWL 1 DL and Full respectively including additional features, such as punning, property chains and qualified cardinality constraints.

The expressiveness of an ontology language influences the complexity of the reasoning algorithms that can be applied to an ontology. Such algorithms can, among others, infer new implicit axioms based on explicitly specified ones, check for satisfiability of classes, compute hierarchies of classes and properties and check consistency of the entire ontology. As there is a trade-off between the richness of an ontology language and the complexity of reasoning tasks, there is also an ongoing quest for ontology languages that strike a balance between these two measures.

As with ontology definition languages, more than a few ontology query languages exist, but the de facto query language for RDF graphs is SPARQL. SPARQL uses RDF graphs expressed in Turtle syntax as query patterns and can return as output variable bindings (`SELECT` queries), RDF graphs (`CONSTRUCT` and `DESCRIBE` queries) or yes/no answers (`ASK` queries). SPARQL is constantly evolving so as to allow update operations as well[8] and it has already been proved to be as expressive as relational algebra [10].

An alternative way of modeling knowledge are rules, which can sometimes express knowledge that can not expressed in OWL. The related W3C Recommendation is RIF[9] which introduces three dialects that share a lot in common with several existing rule languages. The goal is to provide a uniform format for exchange of rules of different kinds. Despite its Recommendation status, RIF has not been widely adopted yet, and instead, the Semantic Web community seems to favour the de facto standard of SWRL[10], which allows rules to be expressed as OWL axioms and thus, be distributed as part of OWL ontologies.

4. A Classification of Approaches

The term *database to ontology mapping* has been loosely used in the related literature, encompassing diverse approaches and solutions to different problems. In this section, we give a classification which will help us categorize and analyze in an orderly manner these approaches. Furthermore, we introduce the descriptive parameters to be used for the presentation of each approach.

Classification schemes and descriptive parameters for database to ontology mapping methods have already been proposed in related work. The distinction between classification criteria and merely descriptive measures is often not clear. Measures that can act as classification criteria should have a finite number of values and ideally, should separate approaches in non overlapping sets. Such restrictions are not necessary for descriptive features, which can sometimes be qualitative by nature instead of quantifiable measures. Table 2 summarizes classification criteria and descriptive parameters proposed in the related literature, despite the fact that some of the works mentioned have a somewhat different (either narrower or broader) scope than ours. As it can be seen from Table 2, there is a certain level of consensus among classification efforts, regarding the most prominent measures characterizing database to ontology mapping approaches.

The taxonomy we propose and which we are going to follow in the course of our analysis is based on some of the criteria mentioned in Table 2. The selection of these criteria is made so as to create a *total* classification of all relevant solutions in *mutually disjoint* classes. In other words, we partition the database to

[7]For more details on OWL 2 Profiles, see `http://www.w3.org/TR/owl2-profiles/`.

[8]SPARQL 1.1 Update: `http://www.w3.org/TR/sparql11-update/`

[9]Rule Interchange Format Overview: `http://www.w3.org/2005/rules/wiki/Overview`

[10]Semantic Web Rule Language: `http://www.w3.org/Submission/SWRL/`

Table 2
Classification criteria and descriptive parameters identified in literature

Work	Classification Criteria	Values	Descriptive Parameters
Auer *et al.* (2009) [14][a]	Automation in the creation of mapping Source of semantics considered Access paradigm Domain reliance	Automatic, Semi-automatic, Manual Existing domain ontologies, Database, Database and User Extract-Transform-Load (ETL), SPARQL, Linked Data General, Dependent	Mapping representation language
Barrasa Rodriguez, Gomez-Perez (2006) [17]	Existence of ontology Architecture Mapping exploitation	Yes (ontology reuse), No (created ad-hoc) Wrapper, Generic engine and declarative definition Massive upgrade (batch), Query driven (on demand)	–
Ghawi, Cullot (2007) [55]	Existence of ontology Complexity of mapping definition Ontology population process Automation in the creation of mapping	Yes, No Complex, Direct Massive dump, Query driven Automatic, Semi-automatic, Manual	Automation in the instance export process
Hellmann *et al.* (2011) [61]	–	–	Data source[b], Data exposition, Data synchronization, Mapping language, Vocabulary reuse, Mapping automation, Requirement of domain ontology, Existence of GUI
Konstantinou, Spanos, Mitrou (2008) [72]	Existence of ontology Automation in the creation of mapping Ontology development	Yes, No Automatic, Semi-automatic, Manual Structure driven, Semantics driven	Ontology language, RDBMS supported, Semantic query language, Database components mapped, Availability of consistency checks, User interaction
Sahoo *et al.* (2009) [105]	Query implementation Data integration	*Same as in Auer et al. (2009) with the addition of:* SPARQL, SPARQL→SQL Yes, No	Mapping accessibility, Application domain
Sequeda *et al.* (2009) [111][c]	–	–	Correlation of primary and foreign keys, OWL and RDFS elements mapped
Zhao, Chang (2007) [129]	Database schema analysis	Yes, No	Purpose, Input, Output, Correlation analysis of database schema elements, Consideration of database instance, application source code and other sources

[a] In this work, approaches are also divided in 4 categories: a) alignment, b) database mining, c) integration approaches and d) mapping languages/servers.
[b] Apart from relational data sources, this report reviews knowledge extraction from XML and CSV data sources as well.
[c] This report investigates only approaches that create a new ontology by reverse engineering the database schema.

ontology mapping problem space in distinct categories containing uniform approaches. The few exceptions we come across are customizable software tools that incorporate multiple workflows, with each one falling under a different category.

We categorize solutions to the database to ontology mapping problem to the classes shown in Figure 1. Next to each class, we append the applicable motivations and benefits, as mentioned in Section 2. Association of a specific benefit to a given class denotes that there is at least one work in this class mentioning this benefit.

The first major division of approaches is based on whether an existing ontology is *required* for the performance of a method. Therefore, the first classification criterion we employ is the **existence of an ontology** as a requirement for the mapping process, distinguishing among methods that establish mappings between a given relational database and a given existing ontology (presented in Section 6) and methods that create a new ontology from a given relational database (presented in Section 5). In the former case, the ontology to be mapped to the database should model a domain that is compatible with the domain of the database contents in order to be able to define meaningful mappings. This

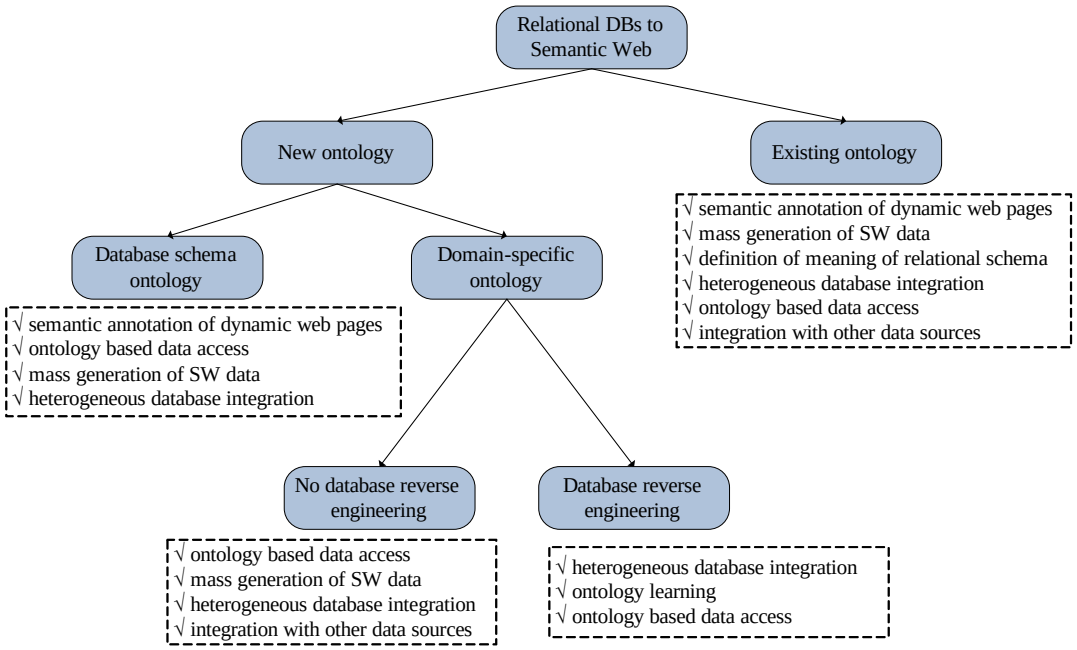

Fig. 1. Classification of approaches.

is the reason why the ontology, in this case, is usually selected by a human user who is aware of the nature of data contained in the database. In the case of methods creating a new ontology, the existence of a domain ontology is not a prerequisite, including use cases where an ontology for the domain covered by the database is not available yet or even where the human user is not familiar with the domain of the database and relies on the mapping process to discover the semantics of the database contents.

The class of methods that create a new ontology is further subdivided in two subclasses, depending on the **domain of the generated ontology**. On the one hand, there are approaches (presented in Section 5.1) that develop an ontology with domain the relational model. The generated ontology consists of concepts and relationships that reflect the constructs of the relational model, mirroring the structure of the input relational database. We will call such an ontology a *"database schema ontology"*. Since the generated ontology does not implicate any other knowledge domain, these approaches are mainly automatic relying on the input database schema alone. On the other hand, there are methods that generate domain-specific ontologies (presented in Section 5.2), depending on the domain described by the contents of the input database.

The last classification criterion we consider is the **application of database reverse engineering** techniques in the process of creating a new domain-specific ontology. Approaches that apply reverse engineering try to recover the initial conceptual schema from the relational schema and translate it to an ontology expressed in a target language (see Section 5.2.2). On the other hand, there are methods that, instead of reverse engineering techniques, apply few basic translation rules from the relational to the RDF model (detailed in Section 5.2.1) and/or rely on the human expert for the definition of complex mappings and the enrichment of the generated ontology model.

By inspection of Figure 1, we can also see that, with a few exceptions, there is not significant correlation between the taxonomy classes and the motivations and benefits presented in Section 2. This happens because, as we argued above, this taxonomy categorizes approaches based on the *nature* of the mapping and the *techniques applied* to establish the mapping. On the contrary, most benefits state the *applications* of the already established mappings (e.g. database integration, web page annotation), which do not depend on the mapping process details. Notable exceptions to the above observation are the ontology learning and definition of semantics of database contents motivations. The former, by definition, calls for approaches that produce an ontology by analyzing a re-

lational database, while the latter calls for approaches that ground the meaning of a relational database to clearly defined concepts and relationships of an existing ontology.

Given the taxonomy of Figure 1 and the three classification criteria employed in it, we choose among the rest of the features and criteria mentioned in Table 2 the most significant ones as descriptive parameters for the approaches reviewed in this paper. The classification criteria and descriptive parameters adopted in this paper are shown in Figure 2 and elaborated in the following:

Level of automation. This variable denotes the extent of human user involvement in the process. Possible values for it are *automatic*, *semi-automatic* and *manual*. Automatic methods do not involve the human user in the process, while semi-automatic ones require some input from the user. This input may be necessary for the operation of the process, constituting an integral part of the algorithm, or it may be optional feedback for the validation and enrichment of the result of the mapping process. In manual approaches, the mapping is defined in its entirety by the human user without any feedback or suggestions from the application. In some cases, automation level is a characteristic that is common among approaches within the same taxonomy class. For instance, methods that create a new database schema ontology are purely automatic, while the majority of approaches that map a relational database to an existing ontology are manual, given the complexity of the discovery of correspondences between the two.

Data accessibility. Data accessibility describes how the mapping result is accessed. This variable is also known in the literature as access paradigm, mapping implementation or data exposition. Possible values for this variable are ETL (Extract, Transform, Load), SPARQL or another ontology query language and Linked Data. ETL means that the result of the mapping process, be it an entire new ontology or a group of RDF statements referencing existing ontologies, is generated and stored as a whole in an external storage medium (often a triple store but it can be an external file as well). In this case, the mapping result is said to be *materialized*. Other terms used for ETL is batch transformation or massive dump. An alternative way of accessing the result of the mapping process is through SPARQL or some alternative semantic query language. In such a case, only a part (i.e. the answer to the query) of the entire mapping result is accessed and no additional storage medium is required since the database contents are not replicated in a materialized group of RDF statements. The semantic query is rewritten to an SQL query, which is executed and its results are transformed back as a response to the semantic query. This access mode is also known as query driven or on demand and characterizes ontology-based data access approaches. Finally, the Linked Data value means that the result of the mapping is published according to the Linked Data principles: all URIs use the HTTP scheme and, when dereferenced, provide useful information for the resource they identify[11].

The data accessibility feature is strongly related to the *data synchronization* measure, according to which, methods are categorized in static and dynamic ones, depending on whether the mapping is executed only once or on every incoming user query, respectively. We believe that the inclusion of data synchronization as an additional descriptive feature would be redundant, since the value of data accessibility uniquely determines whether a method maintains static or dynamic mappings. Hence, an ETL value signifies static mappings, while SPARQL and Linked Data values denote dynamic mappings.

Mapping language. This feature refers to the language in which the mapping is represented. The values for this feature vary considerably, given that, until recently, no standard representation language existed for this purpose and most approaches used proprietary formats. W3C recognized this gap and proposed R2RML[12] (RDB to RDF Mapping Language) as a representation language for relational database to ontology mappings. As R2RML is currently under development, it is expected to be adopted by software tools in the near future. The mapping language feature is applicable only to methods that need to reuse the mapping. A counterexample is the class of ontology learning methods that create a new domain ontology. Methods of this class usually do not represent the mapping in any form, since their intention is simply the generation of a new domain ontology, having no interest in storing correspondences with database schema elements.

Ontology language. This parameter states the language on which the involved ontology is expressed.

[11] The use of the term Linked Data may be slightly misleading, as both the ETL export of RDF data dumps and the SPARQL access scheme are considered alternative ways of Linked Data publishing [60]. Nevertheless, we follow the convention of [14,61,105], where the term Linked Data denotes an RDF access scheme based on HTTP dereferenceable URIs.

[12] R2RML Specification: `http://www.w3.org/TR/r2rml/`

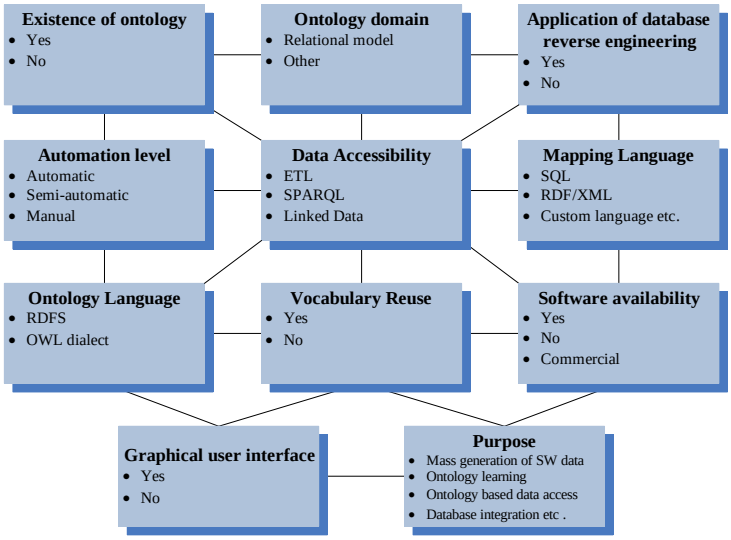

Fig. 2. Classification criteria and descriptive parameters used.

Depending on the kind of approach, it may refer to the language of the ontology generated by the approach or to the language of the existing ontology required. Since the majority of methods reviewed implicate Semantic Web ontologies, the values for this feature range from RDFS to all flavours of OWL. When the expressiveness of the OWL language used is not clear from the description of the method, no specific profile or species is mentioned.

Vocabulary reuse. This parameter states whether a relational database can be mapped to more than one existing ontologies or not. This is mainly true for manual approaches where the human user is usually free to reuse terms from existing ontologies at will. Reusing terms from popular vocabularies is one of the prerequisites for Semantic Web success and hence, is an important aspect for database to ontology mapping approaches. Of course, methods allowing for vocabulary reuse do not *enforce* the user to do so, thus this variable should not be mistaken with the existence of ontology *requirement* that was used as a classification criterion in Figure 1. Once again, values of this parameter may be common among all approaches within a certain class, e.g. methods generating a new database schema ontology do not directly allow for vocabulary reuse, since the terms used are specified a priori by the method and are not subject to change.

Software availability. This variable denotes whether an approach is accompanied with freely accessible software, thus separating purely theoretical methods or unreleased research prototypes from approaches providing practical solutions to the database to ontology mapping problem. Therefore, this feature serves (together with the *purpose* feature) as an index of the available software that should be considered from a potential user seeking support for a particular problem. Commercial software, which is not offering freely its full functionality to the general public, is indicated appropriately.

Graphical user interface. This feature applies to approaches accompanied with an accessible software implementation (with the exception of unavailable research prototypes that have been stated to have a user interface) and shows whether the user can interact with the system via a graphical user interface.This parameter is significant especially for casual users or even database experts who may not be familiar with Semantic Web technologies but still want to bring their database into Semantic Web. A graphical interface that guides the end user through the various steps of the mapping process and provides suggestions is considered as essential for inexperienced users.

Purpose. This is the main motivation stated by the authors of each approach. This is not to say that the motivation stated is the only applicable and that the approach cannot be used in some other context. Usually, all the benefits and motivations showcased in Figure 1 apply to all tools and approaches of the same category.

We note once again that any of the above measures could have been considered as a criterion for the definition of an alternative taxonomy. However, we argue that the criteria selected lead to a partition of the

entire problem space in clusters of approaches that present increased intra-class similarity. For example, some works [14,17,55,105] categorize methods based on the data accessibility criterion to static and dynamic ones. This segregation is not complete though, as there are methods that restrict themselves in finding correspondences between a relational database and an ontology without migrating data to the ontology nor offering access to the database through the ontology. This was the main reason behind the omission of this distinction in the classification of Figure 1. The values of the above descriptive parameters for all approaches mentioned in this paper are presented in Table 8.

Summing up the categories of methods recognized, we will deal with the following, in the next sections of the paper:

1. Creation of database schema ontology (Section 5.1)
2. Creation of domain-specific ontology without database reverse engineering (Section 5.2.1)
3. Creation of domain-specific ontology with database reverse engineering (Section 5.2.2)
4. Definition or discovery of mappings between a relational database and an existing ontology (Section 6)

5. Ontology Creation from Existing Database

In this section, we deal with the issue of generating a new ontology from a relational database and optionally populating the former with RDF data that are extracted from the latter. Schematically, this process is roughly shown on Figure 3. The mapping engine interfaces with the database to extract the necessary information and, according to manually defined mappings or internal heuristic rules, generates an ontology which can also be viewed as a syntactic equivalent of an RDF graph. This RDF graph can be accessed in three ways, as described in Section 4: ETL, SPARQL or Linked Data (lines 1, 3 and 2 in Figure 3, respectively). In ETL, the RDF graph can be stored in a file or in a persistent triple store and be accessed directly from there, while in SPARQL and Linked Data access modes, the RDF graph is generated on the fly from the contents residing in the database. In these modes, a SPARQL query or an HTTP request respectively are transformed to an appropriate SQL query. Finally, the mapping engine retrieves the results of this SQL query (this is not depicted in Figure 3) and formulates a SPARQL solution response or an appropriate HTTP response, depending on the occasion.

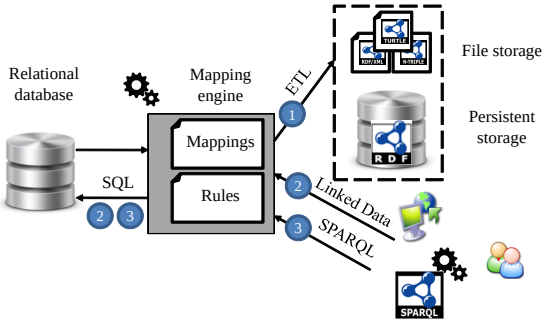

Fig. 3. Generation of ontology from a relational database.

Before we delve into the different categories of ontology creation approaches, we describe as briefly as possible an elementary transformation method of a relational database to an RDF graph proposed by Tim Berners-Lee [22] that, albeit simple, has served as a reference point for several ontology creation methods. This method has been further extended [19] so as to produce an RDFS ontology and is also informally known as "table-to-class, column-to-predicate", but for brevity we are going to refer to it as the *basic approach* throughout this paper. According to the basic approach:

1. Every relation R maps to an RDFS class $C(R)$.
2. Every tuple of a relation R maps to an RDF node of type $C(R)$.
3. Every attribute att of a relation maps to an RDF property $P(att)$.
4. 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 $R[t]$.

As subjects and predicates of RDF statements must be identified with a URI, it is important to come up with a URI generation scheme that guarantees uniqueness for distinct database schema elements. As expected, there is not a universal URI generation scheme, but most of them follow a hierarchical pattern, much like the one presented in Table 3, where for convenience multi-column primary keys are not considered. An important property that the URI generation scheme must possess and is indispensable for SPARQL and Linked Data access is reversibility, i.e. every generated URI should contain enough information to recognize the database element or value it identifies.

The basic approach is generic enough to be applied to any relational database instance and to a large de-

Table 3
Typical 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 the value of the primary key for a given tuple of a relation.

gree automatic, since it does not require any input from the human user, except from the base URI. However, it accounts for a very crude export of relational databases to RDFS ontologies that does not permit more complex mappings nor does it support more expressive ontology languages. Moreover, the resulting ontology looks a lot like a copy of the database relational schema as all relations are translated to RDFS classes, even the ones which are mere artifacts of the logical database design (e.g. relations representing a many-to-many relationship between two entities). Other notable shortcomings of the 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 [31]. Nevertheless, as already mentioned, several methods refine and build on the basic approach, by allowing for more complex mappings and discovering domain semantics hidden in the database structure. Such methods are presented in the next subsections.

5.1. Generation of a database schema ontology

We first review the class of approaches that create a new database schema ontology. Database schema ontologies have as domain the relational model and do not model the domain that is referred to by the contents of the database. In other words, a database schema ontology contains concepts and relationships pertaining to the relational model, such as relation, attribute, primary key and foreign key and simply mirrors the structure of a relational database schema.

The process of generating such an ontology is entirely automated, since all the information needed for the process is contained in the database schema and additional domain knowledge from a human user or external information sources is not needed. Both ETL and SPARQL based approaches are met in this category regarding data accessibility, while the mapping itself is rarely represented in some form. This is due to the fact that the correspondences between the database schema and the generated ontology are straightforward, as every element of the former is usually mapped to an instance of the respective class in the latter. For example, a relation and each one of its attributes in a database schema are mapped to instances of some *Relation* and *Attribute* classes in the generated ontology respectively, with the *Relation* instance being connected with the *Attribute* instances through an appropriate *hasAttribute* property. The names of classes and property in the above example are of course figurative, as every approach defines its own database schema ontology.

An issue frequently arising when modeling a database schema ontology is the need to represent, apart from the database schema, the schema constraints that are applied to the database contents. In the majority of cases, the motivation is the check of database schema constraints using ontology reasoning engines. This requires not only representation of the database schema but also description of the data itself. The most straightforward and natural design pattern that achieves this represents schema-level database concepts as instances of the database schema ontology classes and data is then represented as instances of the schema layer concepts. This design makes use of more advanced *metamodeling* ontology features. Informally, metamodeling allows an ontology resource to be used at the same time as a class, property and individual. Metamodeling violates the separation restriction between classes, properties, individuals and data values that is enforced in OWL (1/2) DL, the decidable fragment of the OWL language, and leads to OWL Full ontologies, which have been shown to be undecidable [89]. The introduction of the punning feature in OWL 2 DL may allow different usages of the same term, e.g. both as a class and an individual, but the standard OWL 2 DL semantics and reasoners supporting it consider

these different usages as distinct terms and thus, anticipated inferences based on OWL (1/2) Full semantics are missed.

Approaches that make use of metamodeling for building a database schema ontology [95,121] create OWL Full ontologies and do not go as far as to check database constraints with appropriate ontology reasoners. This may be due to the popular misbelief that such a task can be carried out only by reasoners that can compute all OWL Full entailments of an ontology. On the contrary, widely available OWL 2 RL reasoners[13] or first order logic theorem proving systems may suffice for the task [108].

The vast majority of methods in this group map a relational database schema to strictly one ontology and do not reuse terms from external vocabularies, since their domain of interest is the relational model. The obvious disadvantage of this group of approaches is the lack of data semantics in the generated database schema ontology, a fact that makes them only marginally useful for integration with other data sources. Thereby, some approaches complement the database schema ontology with manually defined mappings to external domain-specific ontologies. These ontology mappings may take the form of ontology (*rdfs:subClassOf* and *owl:equivalentClass*) axioms [74,75], SPARQL `CONSTRUCT` queries that output a new RDF graph with terms from known ontologies [96] or SWRL rules that contain terms from the database schema ontology in their body and terms from popular ontologies in their head [53]. Such approaches, even indirectly, manage to capture the semantics of database contents and thus, increase the semantic interoperability among independently developed relational databases.

The most popular and dominant work in this category is **Relational.OWL** [95], which is also reused in other approaches and tools (e.g. ROSEX [43] and DataMaster [93]). Relational.OWL embraces the design pattern commented above and makes use of the OWL Full metamodeling features. Every relation and attribute in the relational model gives rise to the creation of a corresponding instance of meta-classes *Table* and *Column*. Data contained in the database is represented according to the rules of the basic approach: every tuple of a relation is viewed as an instance of a schema representation class, while tuple values as viewed as values of properties-instances of the *Column* class. This lack of separation among classes, properties and individuals makes the produced 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). As with most database schema ontologies, the generation of the ontology is performed automatically, without any user intervention. To overcome the lack of domain semantics in the database schema ontology, the authors also propose the use of Relational.OWL as an intermediate step for the creation of an RDF graph that reuses terms from external ontologies [96]. The RDF graph is produced as the result of SPARQL `CONSTRUCT` queries executed on the Relational.OWL ontology. Apart from standard ETL access, the addition of the RDQuery wrapper system [97] allows for querying a relational database through SPARQL queries using terms from the Relational.OWL ontology.

DataMaster [93] is a tool that exports the schema and the contents of a relational database in an ontology using the Relational.OWL terminology. The novelty though is that it also offers two alternative modeling versions of Relational.OWL that manage to stay within the syntactic bounds of OWL DL. In the first one, the schema representation layer of Relational.OWL is completely dropped and thereby, database relations and attributes are translated to OWL classes and properties respectively (much like in the basic approach). In the second one, it is the data representation layer that is missing, since relations and attributes are translated to instances of the *Table* and *Column* Relational.OWL classes respectively, but the latter are not translated to properties as well, therefore avoiding the metamodeling feature used in Relational.OWL.

ROSEX [43] also uses a slightly modified version of the Relational.OWL ontology to represent the relational schema of a database as an OWL ontology, thus ignoring the data representation layer of the latter. The generated database schema ontology is mapped to a domain-specific ontology that is generated automatically by reverse engineering the database schema (much like approaches of Section 5.2.2 do) and this mapping is used for the rewriting of SPARQL queries expressed over the domain ontology to SQL queries

[13] OWL 2 RL Rules (`http://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules`) offer a partial axiomatization of the OWL 2 Full semantics and thus, can be used for developing incomplete reasoning systems that extract some OWL 2 Full entailments.

expressed over the relational database. Although it is argued that the insertion of the database schema ontology as an intermediate layer between the domain ontology and the database can contribute in tracking changes of the domain ontology as the database schema evolves, it is not clear how this is actually achieved.

An approach similar to Relational.OWL that produces a new database schema OWL ontology is **RDB2ONT** [121]. The classes and properties defined follow the same metamodeling paradigm as in 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-level OWL classes, in an unsuccessful effort to avoid OWL Full. Furthermore, it is not clear whether information from the ER conceptual schema of the database is used in the design of ontology, since the exact cardinalities of relationships are assumed in this work to be given, while it is known that they cannot always be deduced by inspection of the relational schema alone.

Avoidance of metamodeling is successful in the work of Kupfer and colleagues [75], where representation of the database contents is not considered. The output of this approach is an OWL-Lite ontology reflecting the database schema, which is later manually enriched with terms from a domain ontology through the use of standard *rdfs:subClassOf* and *owl:equivalentClass* axioms.

Automapper [53] (now part of the commercial Asio SBRD tool) allows a relational database to be queried through SPARQL using terms from a domain ontology. Despite the fact that it is mainly an ontology-based data access solution, we present it in this section, since it involves a database schema ontology as well. Automapper creates an OWL ontology enhanced with SWRL rules to describe the relational schema and express its constraints, such as the uniqueness of the primary key or attribute datatype restrictions. The generation of the schema ontology is automatic and follows the basic approach of one class per relation and one property per attribute. Automapper is the only tool in this category that stores explicitly the mapping among the database schema and the generated ontology. The mapping is expressed in terms of a mapping ontology heavily influenced by D2RQ [27], which is described in Section 5.2. Once again, the construction of a database schema ontology is supplemented with connections to a domain ontology, this time via SWRL rules. SPARQL queries using terms from the domain ontology are translated to SQL queries, taking into account details of the stored mapping, such as the schema-specific URI generation mechanism.

A similar workflow is described in the work of Dolbear and Hart [48], which focuses on spatial databases. Their automatically generated data ontology is enriched with manually defined ontological expressions that essentially correspond to SQL queries in the underlying database. These complex expressions are then linked through ordinary *rdfs:subClassOf* and *owl:equivalentClass* axioms with an existing domain ontology.

FDR2 [74] diverges considerably from the basic approach: every attribute of a relation is mapped to an RDFS class, a relation corresponds to an RDF list 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 the created database schema ontology with an existing domain ontology are defined manually as *rdfs::subClassOf* and rdfs:subPropertyOf axioms.

An interesting summary of solutions for the transformation of relational tuples and hence relational schemas in RDFS ontologies is given in the work of Lausen [78], with ultimate goal the identification of the optimal scheme allowing for database constraint checking, while staying clear of OWL Full solutions. In this work, four different alternatives are presented. The *tuple-based mapping* is essentially the same as the basic approach, with the only difference being the use of blank nodes for each tuple, instead of a generated URI. The obvious disadvantage of this approach is the encoding of all database values as literals, which ignores the presence of foreign keys and leads to duplication of data in the output RDF graph. This is overcome in *value-based mapping*, where URIs are generated for the values of key attributes and thus, "foreign key" properties can directly reference these values in the output RDF graph. The *URI-based mapping* produces a URI for every tuple of a relation, based on the values of the primary key of the relation, pretty much following the URI scheme of Table 3. Also, since primary key values are encoded in the tuple URI, the pri-

mary key of a relation is not explicitly mapped to an RDF property. The drawback of this type of mapping is the fact that foreign key constraints may also implicitly be encoded as dependencies between URIs in the RDF graph, in case the primary key of a relation is also a foreign key to some other relation, a result that is highly undesirable. The *object-based mapping* sticks to the assignment of URIs to the values of key attributes (as in the case of value-based mapping), but foreign keys are represented as properties linking the blank node identifiers of the tuples being linked via the foreign key relationship. Again, foreign key constraints are difficult to be checked in a straightforward way, since a property path should be followed in the RDF graph in order to ensure for referential integrity. Therefore, the solution adopted by Lausen is an RDFS ontology that employs the value-based mapping approach and uses a custom vocabulary of classes and properties for the denotation of primary and foreign keys. This generated database schema ontology is in fact used to check database constraints via execution of appropriate SPARQL queries in [79].

Levshin [81] focuses on database schema constraint checking as well. His approach constructs an OWL ontology that follows the basic approach, but he also enriches it with vocabulary for database concepts and constraint checking. Special care is taken for null values, which are explicitly marked in the generated OWL ontology, contrary to common practice, which simply ignores database null values during the transformation of database contents in RDF. Not null, primary, foreign key and anti-key[14] constraints are encoded as SWRL rules, which are executed as soon as the target ontology is generated. Then, the source of violation of these constraints, if any, is retrieved through SPARQL queries. This approach is slightly different to the one of Lausen and colleagues [79], in which only SPARQL queries are used for the inspection of constraint violations. Both of these works distinguish constraints to positive and negative ones, depending on whether the presence or the lack of some data causes the violation of the constraint. While Lausen and colleagues propose an extension to SPARQL so as to be able to check the violation of all possible constraints, Levshin succeeds in doing so by definition of two distinct properties (*violates* and *satisfies*) for the two type of constraints and then, by combined use of SWRL rules with SPARQL queries for spotting violations.

The complications arising when considering validation of database schema constraints through ontology lenses stem from two fundamental differences in the semantics of relational databases and description logics ontologies. In database systems, both the closed world assumption (CWA) and the unique name assumption (UNA) hold, while DL ontologies follow the open world assumption (OWA) and do not adopt UNA. This means that, what is not stated explicitly in a relational database instance is known to be false, while in a DL ontology, when something is not explicitly stated it is simply not known whether it holds or not. Furthermore, the lack of UNA in an ontology allows different names to refer to the same individual, in contrast with databases where every instance is identified by a unique name. These differences in semantics between databases and ontologies have been extensively investigated in the works of Motik and colleagues [90] and Sirin and Tao [115], proposing different semantics for the interpretation of ontology axioms that act as integrity constraints. In fact, in [115], it is argued that validation of database integrity constraints can be performed through the execution of SPARQL constraints, therefore giving right to the efforts described in [79,81].

Slight variations of the tuple-based mapping are presented in **CROSS** [35] and in the work of Chatterjee and Krishna [36]. CROSS exports an OWL representation of a relational database schema and acts as a wrapper for the data contained in the database, i.e. it employs ETL access to the schema and query based access to the data. For the data component of the created ontology, CROSS adopts a variant of the basic approach: one class for every database relation is constructed, tuples are translated to instances of the relation's class, and values are translated to literals that are not directly linked to the tuple instances, but through an intermediate blank node instead. Database schema constraints are interpreted as OWL axioms and thus, their validity (with the exception of foreign key constraints) can be checked with application of an OWL reasoner. Chatterjee and Krishna [36] construct an RDF graph that follows the principles of tuple-based mapping and they enrich it with vocabulary pertaining to the relational and ER models. No hints are given though as to how this RDF representation could be used for ensuring the validity of data with respect to database schema constraints.

[14] An anti-key constraint for a number of attributes in a relation is satisfied if there is at least one pair of tuples with identical values for these attributes.

OntoMat-Reverse [125] is a semantic annotation tool for dynamic web pages (now part of the OntoMat-Annotizer tool), supporting two different workflow scenarios: one for the annotation of the web presentation and another for direct annotation of the database schema. While the latter workflow is a typical example of a mapping discovery approach between a database schema and an existing ontology (and is therefore described in Section 6), the former involves the construction of a database schema ontology and is of interest in the current Section. The database schema ontology constructed in this approach is a simple RDFS ontology that represents just the structure of the database. Terms from this ontology are used in RDFS representations of SQL queries that, when executed, return values to be used in dynamically generated web pages and which are also embedded in the web presentation. The manual annotation that is then performed through a graphical interface essentially relates an ontology term to the column of the result set of an SQL query. In this case, the constructed database schema ontology is used as a mere provider of column identifiers for the annotation task, thus one could question its necessity, since simpler solutions could have been equally successful for this task.

The relational schema ontology of **Spyder**[15] is another example of a database schema ontology. Spyder is a tool that outputs a domain-specific ontology according to manually specified mappings and therefore, does not quite fit in this category. However, it is mentioned here because, as an intermediate step, it generates an ontology that describes a database schema. This ontology provides a vocabulary for the database terms that are going to be referred to in the specification of mappings.

In Table 4, we summarize the design choices of the mentioned approaches by explicitly presenting the ontology constructs used for each of the relational database schema elements (approach-specific ontology vocabulary is marked with italics), including schema constraints for attributes, such as domain and not null constraints. We also state whether approaches consider the enrichment of their generated database schema ontology with domain semantics, usually in the form of correspondences with an external domain ontology.

[15]Spyder homepage: `http://www.revelytix.com/content/spyder-beta`

5.2. 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. In this section, we deal with approaches that create domain-specific ontologies, in contrast with the database schema ontologies considered in Section 5.1. These domain-specific ontologies do not contain any concepts or relationships that are related to the relational or ER models but instead, concepts and relationships pertaining to the domain of interest that is described by a relational database instance. Naturally, the expressiveness of the generated ontology largely depends on the amount of domain knowledge incorporated in the procedure. The two main sources of domain knowledge are the human user and the relational database instance itself. This is the reason why we distinguish between approaches that mainly rely on reverse engineering the relational schema of a database for the generation of a domain-specific ontology (but can also accept input from a human user) and approaches that do not perform extensive schema analysis but instead are mainly based on the basic approach and optionally, input from a human expert.

5.2.1. Approaches not performing reverse engineering

In this section, we take a look at approaches that generate a new domain-specific ontology from an existing relational database without performing extensive database schema analysis for the extraction of domain semantics. Approaches of this category mainly use the basic approach for exporting a relational database instance to an RDFS ontology and, in the majority of cases, allow a human user who is familiar with the meaning of the database contents to define semantically sound mappings, expressed in a custom mapping representation language. Approaches in this category are usually accompanied with working software applications and this is one of the reasons why this category of tools is by far the most popular and why several relevant commercial tools also begin to emerge[16].

The automation level of this category of tools varies and depends on the level of involvement of the human user in the process. Most of these tools support all of the three possible workflows: from the fully automated

[16]A notable example is Anzo Connect: `http://www.cambridgesemantics.com/products/anzo_connect`

Table 4
Overview of approaches of Section 5.1

Approach	Relational schema elements						Link with domain ontology
	Relation	Attribute	Tuple	Primary Key	Foreign Key	Attribute Constraints	
Automapper [53]	Class	Datatype property	Instance of a relation's class	SWRL rule	–	*owl:allValuesFrom* and *owl:cardinality* = 1 axioms for domain and not null constraints respectively	Yes
CROSS [35]	Subclass of *Row* class	Datatype property	Instance of *Row* class	Object property	Object property	*rdfs:range* restriction on the attribute's property for domain constraints	No
	Class	Datatype property	Instance of a relation class	–	Instance of *ForeignKey* class	–	No
DataMaster [93][a]	Instance of *Table* class	Instance of *Column* class	–	Instance of *PrimaryKey* class	*references* property	*rdfs:range* restriction on *hasXSDType* property for domain constraints	No
FDR2 [74]	RDF list of attributes' classes	Class	RDF list of values	–	–	–	Yes
Kupfer *et al.* (2006) [75]	Instance of *Relation* class	Instance of *Attribute* class	–	–	*isAssociatedWith* property	*consistsOf* property for domain constraints	Yes
Lausen (2007) [78]	Class	Property	Blank node	Instance of *Key* class	Instance of *FKey* class	–	No
Levshin (2009) [81]	Subclass of *Tuple* class	Datatype property	Instance of a relation's class	SWRL rule	SWRL rules	*rdfs:range* restriction on the attribute's property for domain constraints	No
OntoMat-Reverse [125]	Instance of *Table* class	Instance of *Column* class	–	Instance of *PrimaryKey* class	–	*type* property for domain constraints	Yes
RDB2ONT [121]	Instance of the *Relation* class	Property and instance of the *Attribute* class	Instance of a relation's class	Instance of *PrimaryKeyAttribute* class	*referenced Attribute* property	*isNullable* and *type* properties for not null and domain constraints respectively	Yes
Relational OWL [95]	Class and instance of *Table* meta-class	Datatype property and instance of *Column* meta-class	Instance of a relation's class	Instance of *PrimaryKey* class	*references* property	*rdfs:range* restriction on the attribute's property for domain constraints	Yes [96]
ROSEX [43]	Instance of *Table* class	Instance of *Column* class	–	Instance of *PrimaryKey* class	Instance of *ForeignKey* class	–	Yes
Spyder [88]	Instance of *Table* class	Instance of *Column* class	–	Instance of *PrimaryKey* class	Instance of *ForeignKey* class	*dataType*, *defaultValue* and *nullable* properties for domain, default value and not null constraints respectively	No

[a]DataMaster offers three alternative operation modes, one of them being the same as Relational.OWL and therefore not mentioned in the table.

standard basic approach with or without inspection and fine-tuning of the final result by the human user to manually defined mappings. Regarding data accessibility, all three modes are again met among reviewed approaches with a strong inclination towards SPARQL based data access.

The language used for the representation of the mapping is particularly relevant for methods of this group. Unlike the case of tools outputting a database schema ontology, where the correspondences with database elements are obvious, concepts and relationships of a domain-specific ontology may correspond to arbitrarily complex database expressions. To express these correspondences, a rich mapping representation language that contains the necessary features to cover real world use cases is needed. Typical mapping representation languages are able to specify sets of data from the database as well as transformation on these sets that, eventually, define the form of the resulting RDF graph. It comes as no surprise that, until today, with R2RML not being yet finalized, every tool used to devise its own native mapping language, each with unique syntax and features. This resulted in the lock-in of mappings that were created with a specific tool and could not be freely exchanged between different parties and reused across different mapping engines. It was this multilingualism that has propelled the development of R2RML as a common language for expressing database to ontology mappings. Plans for adoption of R2RML are already under way for some tools and

the number of adopters is about to increase in the near future.

Since the majority of the tools we deal with in this section rely on the basic approach for the generation of an ontology, the output ontology language is usually simple RDFS. The goal of these tools is rather to generate a lightweight ontology that possibly reuses terms from other vocabularies for increased semantic interoperability, than to create a highly expressive ontology structure. Vocabulary reuse is possible in the case of manually defined mappings, but the obvious drawback is the fact that, in order to select the appropriate ontology terms that better describe the semantics of the database contents, the user should also be familiar with popular Semantic Web vocabularies. The main motivation that is driving the tools of this Section is the mass generation of RDF data from existing large quantities of data residing in relational databases, that will in turn allow for easier integration with other heterogeneous data.

D2RQ [27] is one of the most prominent tools in the field of relational database to ontology mapping. It supports both automatic and user-assisted operation modes. In the automatic mode, an RDFS ontology is generated according to the rules of the basic approach and additional rules, common among reverse engineering methodologies, for the translation of foreign keys to object properties and the identification of M:N (many-to-many) relationships. In the manual mode, the contents of the relational database are exported to an RDF graph, according to mappings specified in a custom mapping language also expressed in RDF. A semi-automatic mode is also possible, in case the user builds on the automatically generated mapping in order to modify it at will. D2RQ's mapping language offers great flexibility, since it allows for mapping virtually any subset of a relation in an ontology class and several useful features, such as specification of the URI generation mechanism for ontology individuals or definition of translation schemes for database values. D2RQ supports both ETL and query based access to the data. Therefore, 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 [25]. Vocabulary reuse is also supported in D2RQ through manual editing of the mapping representation files.

OpenLink Virtuoso Universal Server is an integration platform that comes in commercial and open-source flavours and offers an RDF view over a relational database with its **RDF Views** feature [28], that offers similar functionality to D2RQ. That is, it supports both automatic and manual operation modes. In the former, an RDFS ontology is created following the basic approach, while in the latter, a mapping expressed in the proprietary Virtuoso Meta-Schema language is manually defined. This mapping can cover complex mapping cases as well, since 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. ETL, SPARQL based and Linked Data access modes are supported. One downside of both D2RQ and Virtuoso 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 basic approach.

In the same vein, **SquirrelRDF** [109] and **Spyder** create an RDFS view of a relational database, allowing for the execution of SPARQL queries against it. Both tools support all the workflows mentioned earlier, that cover the entire range of automation levels. Their automatic mode is typically based on the basic approach and they define their own mapping representation languages, although very different in their expressiveness. SquirrelRDF's mapping language is extremely lightweight and lacks several essential features, as it only supports mapping relations to RDFS classes and attributes to RDF properties. On the contrary, Spyder's mapping language is much richer, while the tool also supports a fair amount of R2RML features. The main data access scheme for both tools is SPARQL based, however SquirrelRDF does not support certain kinds of SPARQL queries that, when transformed to SQL, prove to be very computationally expensive.

Tether [31] also outputs an RDF graph from a relational database. In this work, several shortcomings of the basic approach are identified, some of them mentioned in the introduction of Section 5, and fixes for these shortcomings are proposed. However, this work is targeted specifically to cultural heritage databases and the majority of the suggestions mentioned depend largely on the database schema under examination and the cultural heritage domain itself. The decision as to whether the proposed suggestions are applicable and meaningful in the database under consideration is to be made by a domain expert.

As the basic approach is an integral part of the workflow of most tools in this category, the RDB2RDF Working Group, which is responsible for the R2RML specification, has been chartered in order to also propose a basic transformation of relational databases to RDFS ontologies, called direct mapping [24]. The direct mapping is supposed to standardize the informal basic approach that has been used so far with slight variations from various tools. Therefore, taking into account the structure of a database, the direct mapping produces an RDF graph that can be later modified to achieve more complex mappings.

Triplify [14] is another notable RDF extraction tool from relational schemas. SQL queries are used to select subsets of the database contents and map them to URIs of ontology classes and properties. The mapping is stored in a configuration file, the modification of which can be performed manually and later enriched so as to reuse terms from existing popular vocabularies. The generated RDF triples can be either materialized or published as Linked Data, thus allowing for dynamic access. Triplify also accommodates for external crawling engines, which ideally would want to know the update time of published RDF statements. To achieve this, Triplify also publishes update logs in RDF that contain all RDF resources that have been updated during a specific time period, which can be configured depending on the frequency of updates. The popularity of the Triplify tool is due to the fact that it offers predefined configurations for the (rather stable) database 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 custom mapping language.

A tool that follows a somehow different rationale is **METAmorphoses** [118]. METAmorphoses uses a two-layer architecture for the production of a materialized RDF graph that can potentially reuse terms from other popular ontologies. Apart from the typical mapping layer, where a mapping is defined in a custom XML-based language, there is also a template layer, where the user can specify in terms of another XML-based language the exact output of the RDF graph by reusing definitions from the mapping layer. This architecture introduces unnecessary complexity for the benefit of reusing mapping definitions, a feature that is achieved much more elegantly in R2RML.

Another approach that needs to be mentioned at this point is **OntoAccess** [63], which also introduces a mapping language, R3M, for the generation of an RDF graph from a relational database. Although R3M shows considerably less features from other more advanced mapping languages, it contains all the necessary information for propagating updates of the RDF graph back to the database. OntoAccess offers data access only through SPARQL and is one of the few approaches that focus on the aspect of updating RDF views and ensuring that database contents are modified accordingly. Nevertheless, it does not allow the selection of an arbitrary subset of database contents like most approaches do, but only supports trivial relation to class and attribute and link relations (representing many-to-many relationships) to property mappings.

A very crude overview of some of the features supported by the mapping definition languages of the tools mentioned in this section is shown at Table 5. Among the features included are:

- *Support for conditional mappings*, allowing for the definition of complex mappings that go beyond the trivial "relation-to-class" case, by enabling the selection of a subset of the tuples of the relation, according to some stated condition.
- *Customization of the URI generation scheme for class individuals*, allowing for the definition of URI patterns that may potentially use values from attributes other than the primary key (therefore, generalizing the basic approach URI generation scheme).
- *Application of transformation functions to attribute values* for the customization of the RDF representation. Such transformations may include string patterns combining two or more database values or more complex SQL functions.
- *Support for creation of classes from attribute values*.
- *Foreign key support*, which interprets foreign key attribute values as properties that link two individuals with each other, instead of simple literals.

A more in-depth comparison of mapping representation languages can be found at [64].

5.2.2. Approaches performing reverse engineering

The tools presented in Section 5.2.1 mainly rely on the human user as a source of domain knowledge and support him on the mapping definition task, by providing an initial elementary transformation often based on the basic approach, that can be later customized. On the contrary, approaches presented in this section consider the relational database as the main source of domain knowledge, complemented with knowledge from other external sources and optionally, a human

Table 5
Comparison of language features for tools of Section 5.2.1

Language	Conditional mappings	URI generation scheme for individuals	Transformation functions for attribute values	Attribute values to classes	Foreign key support
D2RQ [27]	√	√	√	√	√
METAmorphoses [118]	√	–	–	–	√
R2RML	√	√	√	√	√
R3M [63]	–	√	–	–	√
Spyder	√	√	√	√	√
SquirrelRDF [109]	–	–	–	–	–
Virtuoso Meta-Schema Language [28]	√	√	√	√	√

expert as well. Examples of such sources are database queries, data manipulation statements, application programs accessing the database, thesauri, lexical repositories, external ontologies and so on.

The majority of methods in this category gain inspiration from traditional database reverse engineering work. The process of reverse engineering the logical schema of a database to acquire a conceptual one is a well known problem of database systems research, having attracted the interest of many researchers since the early 90s. Since the nature of the problem is rather broad, as it encompasses different logical and conceptual models, the related literature is extremely rich and solutions proposed show considerable variance. Several methodologies for the reverse engineering of a relational database to a conceptual schema that follows the Entity-Relationship or other object-oriented models have been proposed and have later been adapted accordingly for providing solution to the relational database to ontology mapping problem. Still though, there exists no silver bullet for the accurate retrieval of the conceptual schema that underpins a relational database. This is due not only to the unavoidable loss of semantics that takes place in the logical database design phase, as different constructs of the ER model map to elements of the same type in the relational model, but also to the fact that database designers often do not use standard design practices nor do they name relations and attributes in a way that gives insight to their meaning [101]. As a consequence, reverse engineering a relational schema is more of an art and less of a formal algorithm with guaranteed results.

Traditional relational database reverse engineering solutions vary significantly because of the different assumptions they are based on. These assumptions refer to the normal form of the input relational schema, the amount of information that is known with respect to the relational schema, the target model the conceptual schema is expressed in and the external sources of information used. Most methods assume that the input relational schema is on third normal form (3NF), which is the least strict normal form that allows for decomposing relations into relations that preserve some interesting properties [102]. Nevertheless, there are methods that are less restrictive and are able to manage relational schemas in second normal form (2NF) [103]. Moreover, some approaches do not consider as given the candidate, primary and foreign keys of each relation and therefore, try to discover as well functional and inclusion dependencies by analyzing the data of the database (e.g. [6,40]). The knowledge of keys in a relational schema is a very important issue, that greatly simplifies the reverse engineering process. As such, it is indispensable for most approaches (e.g. [67,87]), although exceptions do exist [101,103]. Some methodologies focus on specific aspects of the conceptual model, such as cardinalities of participation constraints in relationships [6] or generalization hierarchies [77], and attempt to identify them. Finally, methods differ with respect to the target output model considered: some extract an EER model (e.g. [40]), while others target object-oriented models (e.g. [20,101]). As hinted earlier, most reverse engineering methods are semi-automatic, as they require validation from a human expert and can use additional information sources, such as queries that are executed against the database or relevant domain thesauri [70].

Despite the variance of the above methodologies, they can all prove quite useful and relevant when considering the generation of a domain ontology from relational databases, since Description Logics – on which OWL is based – bear some similarities with object-oriented models. The approaches mentioned in this category accept as input a relational schema, usually in the form of SQL Data Definition Language (DDL) statements, which contain information about primary and foreign keys as well as domain constraints for relational attributes, and generate a domain ontology by mining information not only from the database instance, but also from other sources of domain knowledge. The generation of the ontology is based on a set of heuristic rules, that essentially try to implement the inverse transform of the logical database design process, sketched in Table 1. The variety of assumptions of traditional reverse engineering approaches that was mentioned before is less evident here, as the vast majority of methodologies imply the existence of a 3NF relational schema with full knowledge of keys and domain constraints, while the output is an ontology usually expressed in OWL or another DL language, in the general case.

Apart from this typical line of work, there are also a couple of other relevant groups of methods. The first one considers an ER model as the input to the ontology creation process, while the second one uses intermediate models for the transition from the relational to the ontology model. In the first case, the input for the process is often not available, as conceptual schemas usually get lost after the logical design phase or simply exist in the minds of database designers. Therefore, the practical utility of these tools is arguable, although they tend to generate more expressive ontologies, compared to approaches that work on a relational schema. This is natural, since an ER model is a conceptual model and most, if not all, of its features can be represented in an OWL ontology. Approaches, such as ER2WO [127], ERONTO [122], ER2OWL [52] and the work of Myroshnichenko and Murphy [92] generate ontologies that cover the entire range of OWL species – from OWL Lite to OWL Full – by applying a set of rules that translate constructs of the ER model to OWL axioms. Thus, these tools do not need to apply reverse engineering in the relational schema and are restricted to a simple model translation. The second group of approaches that deserves a mention uses an intermediate model for the generation of an OWL ontology from a relational database [13,65] and constitutes an odd minority, compared to the large number of approaches that directly create a domain ontology.

Approaches of this category are mainly automatic, since they rely, to a large extent, on a set of general rules that are applied on every possible input relational schema. Of course, the result of this procedure cannot be always correct and represent faithfully the meaning of the relational schema and thereby, a human expert is often involved in the process to validate the resulting ontology and, optionally, enrich it with links to other domain ontologies either manually or semi-automatically by following suggestions produced by some algorithm [82]. In some cases, the discovery of correspondences between the generated ontology and other domain ontologies or lexical databases, such as WordNet, is performed automatically (e.g. in [16] and [68] respectively). An ER model can also be taken into account for the validation of the resulting ontology [4]. However, in contrast with early reverse engineering methods, no approach utilizes queries and SQL DML statements in order to extract more accurate domain ontologies. A summary of the information sources considered by approaches of this category for the generation of a domain ontology is depicted in Table 6.

Regarding data accessibility, all approaches output a materialized ontology. This is due to the fact that the main motivation of this group of approaches is the learning of a new ontology that can be later used to other applications and thus, needs to be materialized. This is also reflected in the absence of mapping representation for the majority of these approaches, since the relational database is used as a knowledge source in this context and the storage of mappings is of little interest as they will not be reused. As far as the ontology language of the resulting ontology, there seems to be consensus about the use of OWL, with very few exceptions [11,84,117]. Vocabulary reuse is not possible in this category of approaches, although some of them, as mentioned before, try to establish links with existing domain ontologies, in order to better ground the terms of the newly generated ontology. A shortcoming that is common among all approaches that apply reverse engineering techniques is the structural similarity of the generated ontology with the database schema.

In contrast with the tools of Section 5.2.1, which export the database contents as RDF graphs and, in fact, create instances of RDFS classes, approaches that perform reverse engineering do not always populate the generated ontology classes with data from the database, thus restricting themselves in the generation

Table 6
List of input information sources for approaches of Section 5.2.2

Approach	Input sources			
	Schema	Data	Other Sources	User
Astrova (2004) [11]	√	√		√
Astrova (2009) [12]	√			
Buccella *et al.* (2004) [29]	√			√
DB2OWL [55]	√			
DM-2-OWL [5]	√			
Jurić, Banek, Skočir [68]	√		WordNet	
Lubyte, Tessaris (2009) [84]	√			√
R2O [116]	√		ER model for validation	
RDBToOnto [34]	√	√		√
ROSEX [43]	√			
Shen *et al.* (2006) [112]	√			
SOAM [82]	√		Lexical repositories	√
SQL2OWL [4]	√	√	ER model for validation	√
L. Stojanovic, N. Stojanovic, Volz (2002) [117]	√			√
Tirmizi, Sequeda, Miranker (2008) [120]	√			

of an ontology TBox. Approaches that do populate the generated ontology with instances tend to materialize them, with the exception of DB2OWL [55] and Ultrawrap [110] tools that offer dynamic SPARQL based access.

The heuristic rules on which the ontology generation are often based on the separation of relations, according to the number of attributes that constitute the primary key, the number of foreign keys and the intersection of primary and foreign keys of a relation. In most cases, these rules are described informally [29, 55,82,112,116,117] and sometimes, only through convenient examples that cannot be generalized [11,12]. Moreover, few methods yield semantically sound results, as they often contain faulty rules that misinterpret the semantics of the relational schema, while in other cases, OWL constraints are misused. Two of the most complete and formalized approaches are SQL2OWL [4] and the work of Tirmizi and colleagues [120], in which a set of Horn rules that covers all possible relation cases is proposed.

As the sets of rules are, to a large extent, repeated across different approaches, we refrain from presenting separately each approach and instead present informally the gist of the most significant categories of rules proposed:

1. **Default rules**. These are the rules describing and extending the basic approach for the case of the OWL model, i.e. a relation maps to an OWL class, non-foreign key attributes are mapped to OWL datatype properties and foreign key attributes are mapped to OWL object properties with domain and range the OWL classes corresponding to the parent and child relations respectively. Moreover, a relation tuple maps to an individual of the respective OWL class.
2. **Hierarchy rules**. Such rules identify hierarchies of classes in the relational schema. According to them, when two relations have their primary keys linked with a foreign key constraint, they are mapped to two OWL classes, the one being a superclass of the other. More specifically, the class that corresponds to the parent relation is superclass of the class that corresponds to the child relation. This kind of rules often conflicts with *fragmentation rules*.
3. **Binary relationships rules**. These rules identify a relation that can be mapped to an OWL object property. The primary key of such a relation is composed of foreign keys to two other relations that are mapped to OWL classes, while the relation itself has no additional attributes. One of these OWL classes constitutes the domain and the other one the range of the generated object property, with the choice being left to the human user who may supervise the procedure. In order to overcome this obstacle, automatic methods often create two inverse object properties. In case a relation has additional attributes, it corresponds to a binary relationship with descriptive attributes. Nevertheless, as this feature is not directly supported in OWL, it is mapped to an OWL class with appropriate datatype properties, much like in the *default rules* case.
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 that contains a foreign key to another relation, which represents

the owner entity. Such relations are still mapped to OWL classes, however the semantics of the identifying relationship that connects a weak entity with its owner is difficult to be specified. Some rules, in fact, choose to interpret the relationship between the two relational as a pair of inverse *hasPart* and *isPartOf* object properties. This kind of rules also conflicts with *multi-valued property rules*.

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 that are mapped to OWL classes. Essentially, such rules are not 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, in fact, describe the same entity. Relations identified are mapped to the same OWL class. The identification relies on the presence of the same primary key in all implicated relations, much like in the rules that discover hierarchies of classes.
7. **Multi-valued property rules**. These rules try to identify relations that act as multi-valued attributes. As there are quite a few ways to encode a multi-valued attribute in a relational schema, it is difficult to recognize such an attribute. This kind of rules often assume that a relation that has a composite primary key containing a foreign key referencing a parent relation can be translated to a datatype property with domain the class that corresponds to the parent relation. As mentioned before, these rules are based on the same assumptions with *weak entity rules*.
8. **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 domain constraints regarding an attribute's datatype. These are usually mapped to appropriate OWL cardinality axioms on datatype properties, inverse functional properties and either global (*rdfs:range* axioms) or local universal quantification axioms (e.g. an *owl:allValuesFrom* axiom applied to a specific OWL class) respectively.
9. **Datatype translation rules**. These rules are essentially translation tables defining correspondences between SQL datatypes on the one hand and XML Schema Types, which are typically used as datatypes in RDF and OWL, on the other. Such correspondences are defined in detail in the SQL standard [2].

The rule categories used by every method performing reverse engineering for the generation of a domain-specific ontology are shown in Table 7, together with a note on whether the created ontology is populated with instances from the database. It should be noted that this categorization of rules is somewhat rough, thus slight differences between approaches (e.g. how approaches handle specific constraints, such as primary key, uniqueness, not null constraints etc.) cannot be revealed in Table 7. A similar, more analytic overview of a limited number of approaches, as well as a rather technical table that gathers the precise RDFS and OWL elements used by each approach can be found at [111]. Approaches that include conflicting rule categories, as the ones pinpointed earlier, often rely on the decision of the human user or introduce further premises in order to differentiate between them. However, the validity of these proposals is arguable, since relational schemas that serve as counterexamples can be easily thought of.

It can be easily seen that the rules mentioned before only investigate the relational database schema and do not consider database contents, as some reverse engineering approaches do by exploiting data correlations in the database instance and employing data mining techniques. Until now, very few methods belonging to this category have incorporated some form of data analysis. Astrova [11] examines correlation among key, non-key attributes and data in key attributes between two relations, in order to extract various types of class hierarchies (e.g. both single and multiple inheritance hierarchies). However, this work is structured around informal convenient examples and its results cannot be generalized.

A much more interesting effort is **RDBToOnto** [34], which is the only tool of this category that proposes a formal algorithm for exploiting 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. This data analysis step is applied to attributes that contain some

Table 7
Rules applied to each approach of Section 5.2.2

Approach	Rule categories									Instance population
	1	2	3	4	5	6	7	8	9	
Astrova (2004) [11]	√	√	√	√	√	√	–	√	–	Yes
Astrova (2009) [12]	√	√	√	√	√	–	–	√	√	Yes
Buccella *et al.* (2004) [29]	√	–	√	√	√	–	–	√	√	No
DB2OWL [55]	√	√	√	–	–	–	–	√	√	Yes
DM-2-OWL [5]	√	√	√	–	–	–	–	√	√	No
Jurić, Banek, Skočir [68]	√	–	√	–	–	–	–	√	√	No
Lubyte, Tessaris (2009) [84]	√	–	√	–	√	–	–	√	–	No
R2O [116]	√	√	√	√	√	–	√	√	–	No
RDBToOnto [34]	√	√	√	–	–	–	–	–	–	Yes
ROSEX [43]	√	–	√	√	–	–	–	–	–	No
Shen *et al.* (2006) [112]	√	√	√	–	–	√	–	√	√	Yes
SOAM [82]	√	√	√	√	√	√	–	√	√	Yes
SQL2OWL [4]	√	√	√	–	√	√	√	√	√	Optional
L. Stojanovic, N. Stojanovic, Volz (2002) [117]	√	√	√	–	√	√	–	√	–	Yes
Tirmizi, Sequeda, Miranker (2008) [120]	√	√	√	–	√	–	–	√	√	Yes

lexical clues in their name, which potentially can classify them as categorical attributes. RDBToOnto combines this data analysis step with some of the most common heuristic rules already mentioned and generates an OWL ontology.

As most of the approaches that belong to this group are based on heuristics, their efficiency and ability to interpret correctly the full meaning of a relational schema is doubtful, since they cannot capture all possible database design practices and guess the original intention of the designer. The most significant problems with this kind of approaches are:

Identification of class hierarchies. 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 relation. 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. However, this is only one of the possible alternative patterns that can be used for expressing generalization hierarchies in the relational model [77]. As we have seen, identification of categorical attributes and analysis of data may provide hints for mining subclasses.

Interpretation of primary keys. Primary keys and generally, attributes with unique values are usually transformed to inverse functional datatype properties. The introduction of such axioms leads to OWL Full ontologies, which seem to intimidate ontology designers. Despite the fact that state of the art reasoners, such as Pellet, support inverse functional datatype properties, some methods, in order to avoid OWL Full, model the primary key of a relation as a separate class and translate the primary key constraint to an inverse functional object property. It is also quite surprising that the OWL 2 *hasKey* property is not considered by any approach for the modeling of primary key attributes, although this may be due to the fact that most methods of this group were proposed before the introduction of OWL 2. The *owl:hasKey* property also provides a natural solution for composite primary keys, the transformation of which has not been mentioned in any of the methods reviewed.

Elicitation of highly expressive ontology axioms. So far, no method exists that is able to map successfully expressive OWL features, such as symmetric and transitive properties or even more advanced OWL 2 constructs like property chains, since these features cannot be elicited by inspection of the database schema alone. Furthermore, among the approaches reviewed, there are divergent opinions on the expression of domain constraints for attributes. Some approaches suggest translating these constraints to *rdfs:range* axioms on the corresponding OWL datatype property, while others suggest the use of universal quantification axioms for a specific class with the *owl:allValuesFrom* construct.

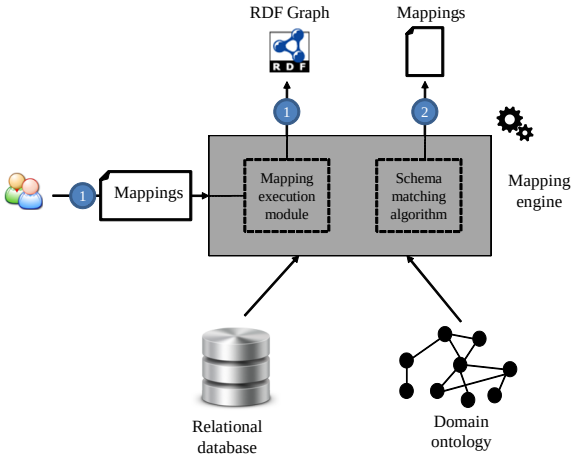

Fig. 4. Mapping an existing ontology to a relational database.

6. Mapping a Database to an Existing Ontology

So far, we have come across tools and methods that do not require a priori knowledge on existing ontologies. In this section, we examine approaches that take as granted the existence of one or more ontologies and either try to establish mappings between a relational database and these ontologies or exploit manually defined mappings for generating an RDF graph that uses terms from these ontologies. These two alternatives are depicted in Figure 4 (lines 2 and 1 respectively). The underlying assumption adopted by all approaches is that the selected ontologies model the same domain as the one modeled by the relational database schema. Therefore, on the one hand, there are approaches that apply schema matching algorithms or augment reverse engineering techniques as the ones discussed in Section 5.2.2 with linguistic similarity measures, in order to discover mappings between a relational database and an appropriately selected ontology. These mappings can then be exploited in various contexts and applications, such as heterogeneous database integration or even database schema matching [7,49]. On the other hand, there are tools that share quite a lot with tools of Section 5.2.1, receiving user-defined mappings between the database schema and one or more existing ontologies and generating an RDF graph that essentially populates ontologies with instance data from the database.

Given that the database schema and the ontology are developed independently, they are expected to differ significantly and often, the domains modeled by the two are not identical, but overlapping. Hence, mappings are usually not trivial or straightforward and, in the general case, they are expressed as complex expressions and formulas. Thus, in any case, tools should be able to discover and express complex mappings, with the first task being particularly challenging and very hard to be automated. As a consequence, the vast majority of approaches in this category are either manual or semi-automatic. In the latter case, some user input is required by the matching algorithm, usually in the form of initial simple correspondences between elements of the database schema and the ontology. A purely automated procedure would need to make oversimplifying assumptions on the lexical proximity of corresponding element names in the database schema and the ontology that are not always true. Such assumptions tend to overestimate the overall efficiency of mapping discovery methods, as for example in MARSON [66] and OntoMat-Reverse [125].

As far as data accessibility is concerned, unlike all categories of tools encountered so far, it is not applicable to approaches that simply output a set of mapping expressions (line 2 in Figure 4), an observation that prevented us from including data accessibility as a classification criterion in Figure 1. For such tools, the main aim is the generation of mappings and thereby, they do not offer RDF dumps of database contents or semantic query based access. These tools left aside, this category supports all three data access scenarios.

Regarding the mapping language parameter, although reusability of mappings is important for this category of approaches, the same comment can be made as in Section 5.2.1, i.e. every tool uses its own proprietary mapping language. Again, expectations are that the level of adoption of R2RML will increase in the near future. All approaches support OWL ontologies, while vocabulary reuse is not possible for some tools that require strictly one ontology to carry out the mapping process. That being said, the vocabulary reuse parameter is also not applicable to approaches that merely discover mappings between a relational database and an ontology, for the same reason as in data accessibility. Another distinguishing feature of tools in this category is the presence of a graphical user interface in the majority of cases. Usually, the interface allows the user to either graphically define the mappings between the database and the ontology, supported by the visualization of their structure or validate automatically discovered mappings.

Starting off with approaches where mappings are manually defined, we mention $\mathbf{R_2O}$ [18], a declarative mapping language that allows a domain expert to define complex mappings between a relational database

instance and one or more existing ontologies. R_2O is an XML-based language that permits the mapping of an arbitrary set of data from the database – as it contains features that can approximate an SQL query – to an ontology class. R_2O's expressiveness is comparable to the most advanced mapping languages discussed in Section 5.2.1, supporting features such as conditional mappings or specification of URI generation schemes. The ODEMapster engine [17] exploits mappings expressed in R_2O in order to create ontology individuals from the contents of the database allowing for either materialized or query-driven access.

Manual mappings between a database schema and a given ontology are defined in **DartGrid** [38] and its successor **Linked Data Mapper** [130]. 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 Local as View (LAV) mappings (database relations expressed as a conjunctive query against the ontology) are generated in the background and stored in an RDF/XML format. DartGrid also allows ontology-based data access, by performing translation of SPARQL queries to SQL, by exploiting the mappings defined. Linked Data Mapper integrates DartGrid's interface with Virtuoso Universal Server, therefore taking advantage of the capabilities and rich features of the latter and enabling, among others, ontology-based data access to the contents of a relational database.

Graphical user interfaces for the manual definition of mappings are also included in the **VisAVis** [71] and **RDOTE** [123] tools. In both approaches, SQL is used as the means for the specification of the data subset that will be mapped to instances of an ontology class. In the case of VisAVis, an SQL query is stored as a literal value of a special purpose datatype property for the corresponding ontology class, while in RDOTE the appropriate SQL query is stored in a configuration file together with possible transformation functions applied to elements of the database. VisAVis allows for query based access, while RDOTE outputs an RDF graph dump that uses classes and properties from the existing ontology. The results of SQL queries are also used for the population of a template of an RDF/XML file in the RDB2Onto tool [76].

The **RDB2OWL** framework [30] adopts a slightly more complex road for the storage and execution of complex mappings between a relational database and one or more existing ontologies. RB2OWL stores the mappings in an external database schema, especially tailored for the purpose. This database schema is, in fact, a mapping meta-schema and contains relations and attributes that store the description of the database schema and ontologies to be mapped as well as the mappings themselves. RDB2OWL allows, similarly to previously mentioned tools, for the selection of a data subset from the database, thus simulating a basic SQL query. Such SQL queries are in fact produced by SQL meta-level queries and executed for the population of the OWL ontologies considered.

Moving on to approaches that discover mappings in a semi-automatic or even automatic way, we first mention the **MAPONTO** tool [8]. MAPONTO is a semi-automatic tool that receives as input from a human expert initial correspondences between relation attributes and properties. The goal of MAPONTO is the extraction of 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. Such mappings define the meaning of database relations in terms of an ontology and can be harnessed in a database integration context. 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 defines 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 (following the typical reverse engineering rules mentioned in Section 5.2.2). By taking into account information initially provided by the user 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 that uses ontology terms. With slight modifications, the same algorithm can be used for investigating the opposite direction, i.e. the discovery of Global as view (GAV) mappings that express concepts of the ontology as conjunctive queries over the database. Such GAV mappings can then be directly exploited for the population of an existing ontology with database instances.

Another interesting approach in this group is the one used in **MARSON** [66]. At first, relations are categorized on groups, according to whether they represent entities or relationships, following the reverse engineering process of Chiang and colleagues [40].

Vectors of coefficients, called *virtual documents*, are then constructed for every relation and attribute of the database schema. The description of a relation takes into account the description of relations connected to it via 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 domain. Likewise, virtual documents for every class and property of the ontology are constructed and similarity between the elements of the database schema and the ontology is computed with pairwise calculation of their identifying vectors' cosine distance. Essentially, the intuition is that the relational schema is interpreted as a graph and some form of elementary graph matching is performed. MARSON adopts the simplest case, in which an element is described only by its name, an obvious shortcoming that eliminates the possibility of matching e.g. a relation named *academic_staff* with a *faculty* class. As a next step, MARSON uses these simple correspondences between relations and OWL classes in order to test the consistency of the attribute-to-property mappings discovered, while it exploits database data and ontology individuals to discover more complex mappings, e.g. finding the most appropriate categorical attribute of a relation that splits it into subsets, which in turn correspond to disjoint OWL subclasses of a superclass. The discovery of the categorical attribute in a relation is accomplished by computing the information gain (a measure known from the information theory field) for every attribute in a relation, a procedure that raises questions regarding the computational efficiency of the approach, as the number of string similarity checks needed for the computation of the information gain is directly related to the data volume of the database. Nevertheless, MARSON is the only effort made so far to completely automatize the process of discovering mappings between a relational database and a given ontology. As such, it relies on the assumption of lexical proximity between names of elements in the database and the ontology during the calculation of similarity coefficients and the assumption of existence of a sufficient number of individuals in the ontology, required for the computation of information gain.

OntoMat-Reverse [125] is part of the OntoMat-Annotizer semantic annotation framework, already mentioned in Section 5.1, where scenario of the indirect annotation of web presentation was described. The direct annotation of the database schema, from which the data of a dynamic web page is retrieved, is the second possible workflow scenario of this framework. To this end, OntoMat-Reverse discovers mappings between the database schema and an existing ontology by complementing the reverse engineering rules of Stojanovic and colleagues [117] with a lexical similarity metric. In essence, it recognizes special structures in the database schema (in much the same way as the methods in Section 5.2.2) and tries to match them lexically with the classes and properties of the existing ontology. The discovery of mappings then trigger the population of the existing ontology with instances from the database. 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. Naturally, validation from an expert is required in the end of the process.

Other tools that rely on lexical similarity measures between database and ontology elements are **RONTO** [94], **D2OMapper** [128] and **AuReLi** [100]. In RONTO, linguistic similarity measures are computed between corresponding structural elements of a database and an ontology. Therefore, relation, non foreign key and foreign key attribute names in the database are compared to class, datatype and object property names in the ontology, respectively, to find the closest lexical matches, which are then validated by the user. A similar matching approach is followed by D2OMapper, which uses the ER2WO [127] translation rules from an ER model to an OWL ontology. However, as these rules present some peculiarities, such as the mapping of binary relationships to OWL classes, it is arguable whether D2OMapper is able to discover mappings between a relational database and an ontology that has *not* been produced by the ER2WO tool. AuReLi, on the other hand, uses several string similarity measures to compare relation attributes with properties of several specified a priori ontologies. Furthermore, AuReLi tries to link database values with ontology individuals by issuing queries to the DBPedia dataset[17]. The results of the algorithm are then presented to the human user for validation. AuReLi is implemented as a D2R Server [25] extension offering Linked Data and SPARQL data access.

In contrast with the string-based techniques described in the previous paragraph, **StdTrip** [106] proposes a structure-based framework that incorporates existing ontology alignment tools, in order to find ontology mappings between a rudimentary ontology that

[17]DBPedia homepage: `http://dbpedia.org/`

is generated from a relational database following the basic approach and a set of given ontologies. As in AuReLi, the results of the ontology alignment algorithm are presented as suggestions to the human user, who selects the most suitable ontology mapping.

We close this section by mentioning **MASTRO** [32], a framework that enables the definition of mappings between a relational database and an ontology and the issue of conjunctive queries expressed in EQL, an SQL-like language. The MASTRO framework is an ontology-based data access tool suite that reformulates a conjunctive query expressed over a *DL-Lite*$_A$ ontology[18] to an SQL query that is evaluated by the relational database. *DL-Lite*$_A$ has been proved to allow for tractable reasoning, while an algorithm has been proposed for the reformulation of a conjunctive query to SQL [98]. This algorithm differs considerably from typical SPARQL to SQL rewriting methods [37,44,50,83] that are employed by other tools offering dynamic data access. MASTRO is also complemented with the OBDA plugin for the popular ontology editor Protégé [99] that allows for the definition of GAV mappings between a database and an ontology via a graphical interface.

7. Discussion

Having gone through all different categories of approaches that pertain to the database to ontology mapping issue, we attempt to summarize and stress some of the main points addressed throughout the paper regarding the *main challenges faced by each group of approaches*. We also argue on why a common evaluation methodology for all proposed solutions is very difficult, if not impossible, to be developed, as some categories of approaches can only be evaluated in a qualitative basis, rather than in terms of an impartial quantified metric. Table 8 lists all database to ontology mapping tools and approaches mentioned throughout the paper, together with values for each of the descriptive parameters identified in Section 4.

Except for the points already discussed in Sections 5 and 6, we point out the limited software availability, as less than half of the approaches are accompanied by publicly available software and even fewer among them are actively maintained and have reached a certain level of maturity. We can also make an informal observation with respect to the *correlation of the level of automation of approaches with their semantic awareness*, i.e. the amount of data semantics they elicit correctly. This happens mainly because a human expert is the most reliable source of semantics of a database schema. Hence, purely manual tools, where mappings are given by the human user, capture by definition the true meaning of the logical schema, whereas purely automatic tools can rarely achieve such levels of successful interpretation. Semi-automatic tools, which interact with the human user, fall somewhere in between, thus we could say that there is some trade-off between the level of automation of an approach and the level of semantic awareness it achieves.

Going into further detail with respect to challenges and efficiency of each category of approaches, we could observe that the **generation of a database schema ontology** (Section 5.1) is rarely a goal per se, and the majority of methods reviewed corroborate this statement. This happens because a database schema ontology is a faithful representation of the relational schema (and in some cases, of the contents as well) of a database. As such, it does not contain any other domain semantics that are usually of interest to a human user or an external application, which would rather abstract the low-level organization details of a database and interact with it in terms of higher-level concepts. That is why most approaches complement the generated ontology with additional axioms that link its terms with concepts from a domain-specific ontology and offer access to the database contents by allowing an external human or software agent to issue queries expressed over this domain-specific ontology. The transformation of the issued semantic queries to SQL queries over the database is performed by taking into account the defined correspondences between the terms of the database schema ontology and the domain-specific one. However, the benefits of adopting a database schema ontology as an in-between layer instead of selecting a direct mapping definition among a database and an ontology have not yet been shown in practice for the task of query rewriting.

On the contrary, reasoning mechanisms are used in cases where an ontological representation of a relational database is employed to check the *validity of database schema constraints*. The challenge with this kind of methods (e.g. [79,81]) is, on the one hand, to bridge the gap between the closed world semantics of database systems with the open world assumption of ontologies and, on the other hand, to achieve an ontological representation that does not increase the

[18] *DL-Lite*$_A$ belongs to the *DL-Lite* family of languages, on which the OWL 2 QL profile is based and is a proper subset of OWL DL.

Table 8: Descriptive parameters of approaches reviewed.

Approach	Level of Automation	Data Accessibility	Mapping Language	Ontology Language	Vocabulary Reuse	Software Availability	Graphical User Interface	Main Purpose
Astrova (2004, 2009) [11,12]	Semi-automatic / Automatic	ETL	–	F-Logic / OWL DL	No	No	No	Ontology learning
AuReLi [100]	Semi-automatic	SPARQL / Linked Data	D2RQ mapping language (custom RDF-based)	OWL	Yes	No	Yes	Ontology-based data access
Automapper (Asio SBRD) [53]	Semi-automatic	SPARQL	custom (RDF-based)	OWL+SWRL	No	Commercial	Yes	Ontology-based data access
Buccella *et al.* (2004) [29]	Semi-automatic	ETL	–	OWL DL	No	No	No	Database integration
CROSS [35]	Semi-automatic	custom access protocol	–	OWL	Yes	Yes	No	Database integration
D2OMapper [128]	Semi-Automatic	–	custom representation (XML file)	OWL DL	No	No	No	Semantic annotation of web pages
D2RQ [27] / D2R Server [25]	Automatic / Manual	ETL/SPARQL / Linked Data	D2RQ mapping language (custom RDF-based)	RDFS	Yes	Yes	No	Ontology-based data access
DartGrid [38]	Manual	SPARQL	custom representation (RDF/XML file)	OWL DL	No	No	Yes	Database integration
DataMaster [93]	Automatic	ETL	–	OWL DL/Full	No	Yes	Yes	Generation of SW data
DB2OWL [55]	Automatic	ETL/SPARQL	R_2O (custom text-based)	OWL DL	No	No	No	Database integration
DM-2-OWL [5]	Automatic	ETL	–	OWL Full	No	No	No	Ontology learning
Dolbear, Hart (2008) [48]	Semi-automatic	SPARQL	–	OWL DL	No	No	No	Spatial database integration
FDR2 [74]	Semi-automatic	ETL	–	RDFS	Yes	No	No	Database interoperability
Kupfer *et al.* (2006) [75]	Semi-automatic	ETL	–	OWL-Lite	Yes	No	No	Database integration
Jurić, Banek, Skočir [68]	Automatic	ETL	–	OWL Full	No	No	No	Ontology learning
Lausen (2007) [78]	Automatic	ETL	–	RDFS	No	No	No	Database integration
Levshin (2009) [81]	Automatic	ETL	–	OWL+SWRL	No	No	No	Constraint validation
Linked Data Mapper [130]	Manual	SPARQL	Virtuoso Meta-Schema Language	OWL	No	Yes	Yes	Database integration

Continued on next page

Approach	Level of Automation	Data Accessibility	Mapping Language	Ontology Language	Vocabulary Reuse	Software Availability	Graphical User Interface	Main Purpose
Lubyte, Tessaris (2009) [84]	Semi-automatic	SPARQL	–	*DLR-DB*	No	No	No	Ontology-based data access
MAPONTO [8]	Semi-Automatic	–	custom representation (XML file)	OWL DL	No	Yes	Yes	Meaning definition of database
MARSON [66]	Automatic	–	custom representation (XML file)	OWL DL	No	No	No	Database interoperability
MASTRO [32] / OBDA plugin for Protégé [99]	Manual	SPARQL	custom representation	*DL-Lite*$_A$ / OWL 2 QL	No	Yes	Yes	Ontology-based data access
METAmorphoses [118]	Manual	ETL	custom (XML-based)	RDF	Yes	Yes	Yes	Generation of SW data
OntoAccess [63]	Manual	SPARQL Update	custom (RDF-based)	RDFS	Yes	Yes	No	Ontology-based data update
OntoMat-Reverse [125] (OntoMat-Annotizer)	Semi-Automatic	ETL/F-Logic	F-Logic	F-Logic	No	Yes	Yes	Semantic annotation of web pages
R2O [116]	Semi-automatic	ETL	–	OWL DL	No	No	No	Data integration
R_2O / ODEMapster [17]	Manual	ETL / custom query language (ODEMQL)	R_2O (custom text-based)	OWL	Yes	Yes	No	Ontology-based data access
RDB2ONT [121]	Automatic	ETL	–	OWL Full	No	No	No	Database integration
RDB2OWL [30]	Manual	ETL	SQL	OWL	Yes	No	No	Generation of SW data
RDBToOnto [34]	Semi-automatic	ETL	–	OWL	No	Yes	Yes	Ontology learning
RDOTE [123]	Manual	ETL	SQL	OWL	Yes	Yes	Yes	Generation of SW data
Relational.OWL [95] / RDQuery [97]	Automatic	ETL/SPARQL	–	OWL Full	No	Yes	Yes	Data exchange in a P2P network
RONTO [94]	Semi-automatic	–	custom representation	RDFS/OWL	No	No	Yes	Generation of SW data
ROSEX [43]	Automatic	SPARQL	custom ontology	OWL DL	No	No	No	Ontology-based data access
Shen *et al.* (2006) [112]	Automatic	ETL	–	OWL DL	No	No	No	Database integration
SOAM [82]	Semi-Automatic	ETL	–	OWL DL	No	No	No	Ontology learning
SQL2OWL [4]	Semi-Automatic	ETL	–	OWL DL	No	No	Yes	Ontology learning
Spyder	Automatic / Manual	ETL/SPARQL	custom (RDF-based) / R2RML	RDFS	Yes	Yes	No	Ontology-based data access
SquirrelRDF [109]	Automatic / Manual	SPARQL	custom (RDF-based)	RDFS	Yes	Yes	No	Ontology-based data access

Continued on next page

Approach	Level of Automation	Data Accessibility	Mapping Language	Ontology Language	Vocabulary Reuse	Software Availability	Graphical User Interface	Main Purpose
StdTrip [106]	Semi-automatic	ETL	custom representation / SQL	OWL	Yes	No	Yes	Generation of SW data
L. Stojanovic, N. Stojanovic, Volz (2002) [117]	Semi-automatic	ETL	–	F-Logic / RDFS	No	No	No	Semantic annotation of web pages
Tether [31]	Semi-Automatic	ETL	–	RDFS	Yes	No	No	Generation of SW data
Tirmizi, Sequeda, Miranker (2008) [120]	Automatic	ETL	–	OWL DL	No	No	No	Generation of SW data
Triplify [14]	Manual	ETL / Linked Data	SQL	RDFS	Yes	Yes	No	Generation of SW data
Ultrawrap [110]	Automatic	SPARQL	–	OWL DL	No	No	No	Ontology-based data access
Virtuoso RDF Views [28]	Automatic / Manual	ETL/SPARQL / Linked Data	Virtuoso Meta-Schema Language	RDFS	Yes	Yes	Yes	Ontology-based data access
VisAVis [71]	Manual	RDQL	SQL	OWL DL	No	No	Yes	Meaning definition of database

complexity of reasoning tasks beyond acceptable levels. Methods dealing with this problem use a variety of technologies, including rules, SPARQL queries as well as the metamodeling and the inverse functional datatype property features of OWL. However, there are strong doubts regarding the practical value of such solutions, given that validation of database schema constraints is already performed efficiently by database management systems, does not constitute an important database performance factor and it is highly unlikely that solutions involving ontology reasoning engines will prove more efficient. In fact, none of these works offers a practical evaluation of their proposals, which seem more like theoretical exercises examining the interactions of the relational model with OWL ontologies. A suitable evaluation methodology would probably need to assess the efficiency of each approach with respect to the number of relations and constraints in a database schema as well as the size of the data in the database. For each test case, it could then calculate the total time needed for constraint validation and compare it with the time needed by a database management system.

Moving on to approaches that **create a new domain ontology** (Section 5), the main challenges faced by tools of this category include the definition of feature-rich mapping languages that allow for a fully customizable export of database contents to RDFS ontologies and for the storage of information required by query rewriting algorithms, the search for efficient query rewriting algorithms, the extraction of domain knowledge from the database schema and the, as automated as possible, enrichment of the generated ontology with knowledge from other relevant domain sources. However, these high-level goals depend on the motivation considered by each approach and usually, cannot be all sought after together. The first two of them are normally faced by tools of Section 5.2.1, the majority of which support dynamic query-based data access, while the third one is the subject of approaches presented in Section 5.2.2.

The definition of new expressive and user-friendly database to ontology mapping languages has been a goal for researchers in the past, but the standardization process of R2RML that is currently under way has already moved the focus away from research on new languages. On the contrary, research on efficient query rewriting algorithms still remains a challenge for **tools aiming at ontology-based data access** and constitutes an important success factor for them. In fact, the emergence of efficient strategies for the transformation of SPARQL queries to equivalent SQL ones (e.g. [37,50]) has led to the consideration of such database to ontology mapping tools as an alternative to triple stores, at least in the case of data that resides in relational databases and is not native RDF. Since relational database management systems, due to their robust optimization strategies, outperform triple stores in the query answering task by factors up to one hundred [26], it is not surprising that, as long as SPARQL to SQL rewriting algorithms do not introduce a large delay, the combination of a relational database with a dynamic access database to ontology mapping tool and will still outperform triple stores. This is true for at least two such tools, namely D2R Server [25] and the RDF Views feature of Virtuoso Universal Server [28], which have been found to perform better than conventional triple stores [26] for large datasets.

Although the *efficiency of database to ontology mapping systems that offer SPARQL data access* can be quantified by calculating the response time to a query, it is not a trivial task to come up with an *objective benchmark procedure* for the fair and thorough comparison of the performance of such systems. Indeed, some of the evaluation methodologies that have been proposed present contradictory results regarding the performance of mapping tools and native triple stores with respect to SPARQL query answering [26,57]. A proper benchmarking methodology for this category of tools usually involves the construction of data generators outputting datasets of various sizes with desired properties and the proposal of sequences of queries of different complexity that are issued against these datasets. The efficiency of a system is then evaluated in terms of the response time for a sequence of these queries or equivalently, in terms of the number of queries that can be executed by the system in a given time period. A procedure like the above is implemented in the Berlin SPARQL Benchmark [26], which is one of the most acclaimed benchmarks for storage systems that expose SPARQL endpoints and, as such, it also includes D2R Server and Virtuoso RDF Views in its tests. Another similar notable benchmark is the SP^{2}Bench SPARQL performance benchmark [107], which can be applied to both triple stores and SPARQL to SQL rewriting systems, although none of the latter have been included in its tests.

Another measure that is more relevant to approaches that materialize the generated RDF graph is the time needed for the production of an RDF statement. Two of the factors that influence this measure is the to-

tal size and the complexity of the RDF graph generated [119]. Therefore, a suitable evaluation strategy would, once again, involve the considerations of RDF graphs of different sizes and complexities. However, this measure is usually of little interest, especially to approaches that are driven by the motivation of learning a domain ontology from a relational database and reusing it in some other application. In such cases, the production time of the ontology is not the key factor for success, since this procedure will usually be performed only once. What is more interesting for the end user and, in the same time, challenging for the tool is the elicitation of domain knowledge from the relational database instance as well as knowledge extraction from additional information sources in case the relational database proves insufficient for the purpose. As far as the above goals are considered, the *efficiency of tools that learn an ontology from a relational database* is very hard to be quantified and measured objectively. Some systems measure the efficiency of their approach by comparing the generated ontology with an a priori known ER model corresponding to the input relational database. The above rationale could be somehow quantified, if pairs of relational database schemas and "gold standard" ontologies were specified, with the latter capturing the intended domain semantics contained in the database. Ideally, these test pairs will need to include relational schemas following non-standard design practices often met in real world applications. The efficiency of an approach would then be dependent on the amount of ontology constructs identified as expected and on the total number of test cases handled correctly.

Such a collection of test cases could also be employed for the *evaluation of the performance of tools that discover mappings between a relational database and an ontology*, presented in Section 6. In fact, a short collection of pairs of database schemas and ontologies has been created for the purpose of the evaluation of the MAPONTO tool [8] and has also been applied in the evaluation of MARSON [66]. This collection of schemas and ontologies together with the related collection gathered in the relational-to-RDF version of THALIA testbed[19] could serve as the basis for the development of a full benchmark that will incorporate more complex real world schemas and ontologies from a variety of subject domains.

As the goal of **mapping discovery tools** is to automatize as much as possible the mapping discovery process, a true challenge for them is to rely less on lexical similarity techniques that pose restrictions on their effectiveness and try to solve the synonym and homonym problems with the aid of appropriate lexical reference sources and techniques that compare the structure of the two models. As far as approaches that support the manual definition of mappings between a database and an existing ontology, also presented in Section 6, the challenges are more or less the same as the ones previously mentioned for the tools of Section 5.2.1, i.e. the search for an expressive mapping language and the efficient query rewriting for the case of tools that support dynamic data access. In addition to these high-level goals, tools of this kind also face the challenge of presenting to the end user an intuitive graphical interface that will allow him to define complex mappings, without requiring him to be familiar with the underlying mapping representation language. This is a very important prerequisite for the widespread use of such tools, that unfortunately is usually not given enough attention, in favour of other more tangible indicators of performance.

From the above, it becomes evident that a non-superficial and thorough analysis of the efficiency of reviewed approaches needs special care and cannot be performed in the context of the current paper. An additional practical difficulty to the ones already mentioned is the lack of a common standard application interface for software implementations, with the exception of tools that expose a SPARQL endpoint, which are easily pluggable to an external benchmark application, regardless of the implementation's programming language. This is not true for other categories of tools, which express their output in various forms, i.e. different ontology languages and/or different mapping representation formats, making their comparison cumbersome. As already stated, the only relevant comparative evaluation available is the Berlin SPARQL Benchmark and we encourage the interested reader to consult [26] for the detailed performance analysis of D2R Server and Virtuoso RDF Views. As far as evaluation of other tool categories is concerned, we gathered at one place[20] all experimental datasets mentioned in the

[19] Relational-to-RDF version of THALIA benchmark: `http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/THALIATestbed`

[20] Test datasets for relational to ontology mapping: `http://orpheus.cn.ece.ntua.gr/rdb-rdf`

literature and used in the evaluation of individual approaches, hoping that they come of use for the evaluation of new relevant methods.

8. Future Directions

In this paper, we tried to present the wealth of research work marrying the worlds of relational databases and Semantic Web. We illustrated the variety of different approaches and identified the main challenges that researchers of this field face as well as proposed solutions. We close this paper by mentioning some problems that have only been lightly touched upon by database to ontology mapping solutions as well as some aspects that need to be considered by future approaches.

1. **Ontology-based data update**. A lot of approaches mentioned offer SPARQL based access to the contents of the database. However, this access is unidirectional. Since the emergence of SPARQL Update that allows update operations on an RDF graph, the idea of issuing SPARQL Update requests that will be transformed to appropriate SQL statements and executed on the underlying relational database has become more and more popular. Some early work has already appeared in the OntoAccess prototype [63] and the extensions on the D2RQ tool, D2RQ/Update[21] and D2RQ++ [104]. However, as SPARQL Update is still under development and its semantics is not yet well defined, there is some ambiguity regarding the transformation of some SPARQL Update statements. Moreover, only basic (relation-to-class and attribute-to-property) mappings have been investigated so far. The issue of updating relational data through SPARQL Update is similar to the classic database view update problem, therefore porting already proposed solutions would contribute significantly in dealing with this issue.
2. **Mapping update**. Database schemas and ontologies constantly evolve to suit the changing application and user needs. Therefore, established mappings between the two should also evolve, instead of being redefined or rediscovered from scratch. This issue is closely related to the previous one, since modifications in either participating model do not simply incur adaptations to the mapping but also cause some necessary changes to the other model as well. So far, only few solutions have been proposed for the case of the unidirectional propagation of database schema changes to a generated ontology [42] and the consequent adaptation of the mapping [9]. The inverse direction, i.e. modification of the database as a result of changes in the ontology has not been investigated thoroughly yet. On a practical note, both database trigger functions and mechanisms like the Link Maintenance Protocol (WOD-LMP) from the Silk framework [124] could prove useful for solutions to this issue.
3. **Generation of Linked Data**. A fair number of approaches support vocabulary reuse, a factor that has always been important for the progress of the Semantic Web, while a few other approaches try to discover automatically the most suitable classes or properties from popular vocabularies that can be mapped to a given database schema. Nonetheless, these efforts are still not adequate for the generation of RDF graphs that can be smoothly integrated in the Linking Open Data (LOD) Cloud[22]. For the generation of true Linked Data, the real world entities that database values represent should be identified and links between them should be established, in contrast with the majority of current methods, which translate database values to RDF literals. Lately, a few interesting tools that handle the transformation of spreadsheets to Linked RDF data by analyzing the content of spreadsheet tables have been presented, with the most notable examples being the RDF extension for Google Refine [85] and T2LD [91]. Techniques as the ones applied in these tools can certainly be adapted to the relational database paradigm.

These aspects, together with the challenges enumerated in Section 7, mark the next steps for database to ontology mapping approaches. Although a lot of ground has been covered during the last decade, it looks like there is definitely some interesting road ahead in order to seamlessly integrate relational databases with the Semantic Web, turning it into reality at last.

[21] D2RQ/Update and D2R/Update Server homepage: `http://d2rqupdate.cs.technion.ac.il/`

[22] LOD Cloud diagram: `http://richard.cyganiak.de/2007/10/lod/`

Acknowledgements

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

References

[1] ISO/IEC 9075-1:2008, SQL Part 1: Framework (SQL/Framework), International Organization for Standardization, 27 January 2009.

[2] ISO/IEC 9075-14:2008, SQL Part 14: XML-Related Specifications (SQL/XML), International Organization for Standardization, 27 January 2009.

[3] S. Abiteboul, R. Hull and V. Vianu, *Foundations of Databases*, Addison-Wesley, New York City, NY, 1st ed., 1995.

[4] N. Alalwan, H. Zedan and F. Siewe, Generating OWL Ontology for Database Integration, in P. Dini, J. Hendler, J. Noll et al., eds., *Proceedings of the Third International Conference on Advances in Semantic Processing (SEMAPRO 2009)*, pp. 22–31, IEEE, 2009.

[5] K. M. Albarrak and E. H. Sibley, Translating Relational & Object-Relational Database Models into OWL Models, in S. Rubin and S.-C. Chen, eds., *Proceedings of the 2009 IEEE International Conference on Information Reuse & Integration (IRI 2009)*, pp. 336–341, IEEE, 2009.

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

[7] Y. An, A. Borgida, R. J. Miller and J. Mylopoulos, A Semantic Approach to Discovering Schema Mapping Expressions, in R. Chirkova and V. Oria, eds., *Proceeding of 2007 IEEE 23rd International Conference on Data Engineering (ICDE 2007)*, pp. 206–215, IEEE, 2007.

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

[9] Y. An, X. Hu and I.-Y. Song, Round-Trip Engineering for Maintaining Conceptual-Relational Mappings, in Z. Bellahsène and M. Léonard, eds., *Advanced Information Systems Engineering: 20th International Conference (CAiSE 2008)*, *Lecture Notes on Computer Science*, vol. 5074, pp. 296–311, Springer, 2008.

[10] R. Angles and C. Gutierrez, The Expressive Power of SPARQL, in A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin and K. Thirunarayan, eds., *The Semantic Web - ISWC 2008: 7th International Semantic Web Conference*, *Lecture Notes in Computer Science*, vol. 5318, pp. 114–129, Springer, 2008.

[11] I. Astrova, Reverse Engineering of Relational Databases to Ontologies, in C. J. Bussler, J. Davies, D. Fensel and R. Studer, eds., *The Semantic Web: Research and Applications: First European Semantic Web Symposium (ESWS 2004)*, *Lecture Notes in Computer Science*, vol. 3053, pp. 327–341, Springer, 2004.

[12] I. Astrova, Rules for Mapping SQL Relational Databases to OWL Ontologies, in M.-A. Sicilia and M. D. Lytras, eds., *Metadata and Semantics*, pp. 415–424, Springer, 2009.

[13] P. Atzeni, S. Paolozzi and P. Del Nostro, Ontologies and Databases: Going Back and Forth, in *Proceedings of the 4th International VLDB Workshop on Ontology-based Techniques for Databases in Information Systems and Knowledge Systems (ODBIS 2008)*, pp. 9 – 16, 2008.

[14] 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 (WWW'09)*, pp. 621–630, ACM, 2009.

[15] F. Baader, D. L. McGuinness, P. F. Patel-Schneider and D. Nardi, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2nd ed., 2007.

[16] M. Baglioni, M. V. Masserotti, C. Renso and L. Spinsanti, Building Geospatial Ontologies from Geographical Databases, in F. Fonseca, M. A. Rodríguez and S. Levashkin, eds., *GeoSpatial Semantics: Second International Conference (GeoS 2007)*, *Lecture Notes in Computer Science*, vol. 4853, pp. 195–209, Springer, 2007.

[17] J. Barrasa - Rodriguez and A. Gómez-Pérez, Upgrading Relational Legacy Data to the Semantic Web, in *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*, pp. 1069–1070, ACM, 2006.

[18] J. Barrasa, O. Corcho and A. Gómez-Pérez, R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language, in *Second International Workshop on Semantic Web and Databases (SWDB 2004)*, 2004.

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

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

[21] C. Ben Necib and J.-C. Freytag, Semantic Query Transformation Using Ontologies, in B. C. Desai and G. Vossen, eds., *Proceedings of 9th International Database Engineering & Application Symposium (IDEAS 2005)*, pp. 187–199, IEEE, 2005.

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

[23] T. Berners-Lee, Semantic Web Road map, available at: `http://www.w3.org/DesignIssues/Semantic.html`, 1998.

[24] A. Bertails and E. G. Prud'hommeaux, Interpreting Relational Databases in the RDF Domain, in M. A. Musen and O. Corcho, eds., *Proceedings of the 2011 Knowledge Capture Conference (K-CAP 2011)*, pp. 129–135, ACM, 2011.

[25] C. Bizer and R. Cyganiak, D2R Server - Publishing Relational Databases on the Semantic Web, *poster in 5th International Semantic Web Conference (ISWC 2006)*, 2006.

[26] C. Bizer and A. Schultz, The Berlin SPARQL Benchmark, *International Journal On Semantic Web and Information Sys-*

tems, **5**(2), pp. 1–24, 2009.

[27] C. Bizer and A. Seaborne, D2RQ - Treating non-RDF Databases as Virtual RDF Graphs, *poster in 3rd International Semantic Web Conference (ISWC 2004)*, 2004.

[28] C. Blakeley, Virtuoso RDF Views Getting Started Guide, available at: `http://www.openlinksw.co.uk/virtuoso/Whitepapers/pdf/Virtuoso_SQL_to_RDF_Mapping.pdf`, OpenLink Software, 2007.

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

[30] G. Būmans and K. Čerāns, RDB2OWL : a Practical Approach for Transforming RDB Data into RDF/OWL, in A. Paschke, N. Henze and T. Pellegrini, eds., *Proceedings of the 6th International Conference on Semantic Systems (I-SEMANTICS 2010)*, ACM, 2010.

[31] K. Byrne, Having Triplets - Holding Cultural Data as RDF, in M. Larson, K. Fernie, O. J. and J. Cigarran, eds., *Proceedings of the ECDL 2008 Workshop on Information Access to Cultural Heritage*, 2008.

[32] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi and D. F. Savo, The MASTRO System for Ontology-based Data Access, *Semantic Web Journal*, **2**(1), pp. 43–53, 2011.

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

[34] F. Cerbah, Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies, in Y. Li, G. Pasi, C. Zhang, N. Cercone and L. Cao, eds., *Proceedings of 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops(WI-IAT 2008)*, pp. 553–557, IEEE, 2008.

[35] P.-A. Champin, G.-J. Houben and P. Thiran, CROSS: an OWL Wrapper for Reasoning on Relational Databases, in C. Parent, K.-D. Schewe, V. C. Storey and B. Thalheim, eds., *Conceptual Modeling - ER 2007: 26th International Conference on Conceptual Modeling*, *Lecture Notes in Computer Science*, vol. 4801, pp. 502–517, Springer, 2007.

[36] N. Chatterjee and M. Krishna, Semantic Integration of Heterogeneous Databases on the Web, in B. B. Bhattacharyya, C. A. Murthy, B. B. Chaudhuri and B. Chanda, eds., *International Conference on Computing: Theory and Applications (ICCTA 2007)*, pp. 325–329, IEEE, 2007.

[37] A. Chebotko, S. Lu and F. Fotouhi, Semantics Preserving SPARQL-to-SQL Translation, *Data & Knowledge Engineering*, **68**(10), pp. 973–1000, 2009.

[38] H. Chen, Z. Wu, Y. Mao and G. Zheng, DartGrid: a Semantic Infrastructure for Building Database Grid Applications, *Concurrency and Computation: Practice and Experience*, **18**(14), pp. 1811–1828, 2006.

[39] 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.

[40] 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.

[41] E. F. Codd, A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, **13**(6), pp. 377–387, 1970.

[42] O. Curé and R. Squelbut, A Database Trigger Strategy to Maintain Knowledge Bases Developed via Data Migration, in C. Bento, A. Cardoso and G. Dias, eds., *Progress in Artificial Intelligence: 12th Portuguese Conference on Artificial Intelligence (EPIA 2005)*, *Lecture Notes in Computer Science*, vol. 3808, pp. 206–217, Springer, 2005.

[43] C. Curino, G. Orsi, E. Panigati and L. Tanca, Accessing and Documenting Relational Databases through OWL Ontologies, in T. Andreasen, R. R. Yager, H. Bulskov, H. Christiansen and H. Legind Larsen, eds., *Flexible Query Answering Systems: 8th International Conference (FQAS 2009)*, *Lecture Notes in Computer Science*, vol. 5822, pp. 431–442, Springer, 2009.

[44] R. Cyganiak, A Relational Algebra for SPARQL, Technical Report HPL-2005-170, Hewlett-Packard, 2005.

[45] S. Das and J. Srinivasan, Database Technologies for RDF, in S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset and R. A. Schmidt, eds., *Reasoning Web. Semantic Technologies for Information Systems: 5th International Summer School 2009*, *Lecture Notes in Computer Science*, vol. 5689, pp. 205–221, Springer, 2009.

[46] J. de Bruijn, F. Martin-Recuerda, D. Manov and M. Ehrig, State-of-the-art Survey on Ontology Merging and Aligning, SEKT Project, Deliverable D4.2.1, 2004.

[47] M. del Mar Roldan-Garcia and J. F. Aldana-Montes, A Survey on Disk Oriented Querying and Reasoning on the Semantic Web, in R. S. Barga and X. Zhou, eds., *Proceedings of 22nd International Conference on Data Engineering Workshops (ICDEW'06)*, IEEE, 2006.

[48] C. Dolbear and G. Hart, Ontological Bridge Building - Using Ontologies to Merge Spatial Datasets, in D. L. McGuinness, P. Fox and B. Brodaric, eds., *Semantic Scientific Knowledge Integration: AAAI Spring Symposium (AAAI/SSS Workshop)*, pp. 15–20, 2008.

[49] E. Dragut and R. Lawrence, Composing Mappings between Schemas using a Reference Ontology, in R. Meersman and Z. Tari, eds., *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, *Lecture Notes in Computer Science*, vol. 3290, pp. 783–800, Springer, 2004.

[50] B. Elliott, E. Cheng, C. Thomas-Ogbuji and Z. M. Ozsoyoglu, A Complete Translation from SPARQL into Efficient SQL, in B. C. Desai, ed., *Proceedings of the 2009 International Database Engineering & Applications Symposium (IDEAS '09)*, pp. 31–42, ACM, 2009.

[51] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc., San Francisco, CA, USA, 6th ed., 2010.

[52] M. Fahad, ER2OWL: Generating OWL Ontology from ER Diagram, in Z. Shi, E. Mercier-Laurent and D. Leake, eds., *Intelligent Information Processing IV: 5th IFIP International Conference on Intelligent Information Processing*, pp. 28–37, Springer, 2008.

[53] M. Fisher, M. Dean and G. Joiner, Use of OWL and SWRL for Semantic Relational Database Translation, in K. Clark and P. F. Patel-Schneider, eds., *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, 2008.

[54] J. Geller, S. A. Chun and Y. J. An, Toward the Semantic Deep Web, *Computer*, **41**(9), pp. 95–97, 2008.

[55] R. Ghawi and N. Cullot, Database-to-Ontology Mapping Generation for Semantic Interoperability, in *3rd International*

Workshop on Database Interoperability (InterDB 2007), held in conjunction with VLDB 2007, 2007.

[56] A. Gomez-Perez, O. Corcho-Garcia and M. Fernandez-Lopez, *Ontological Engineering*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st ed., 2003.

[57] A. J. G. Gray, N. Gray and I. Ounis, Can RDB2RDF Tools Feasibly Expose Large Science Archives for Data Integration?, in L. Aroyo, P. Traverso, F. Ciravegna et al., eds., *The Semantic Web: Research and Applications: 6th European Semantic Web Conference (ESWC 2009)*, *Lecture Notes in Computer Science*, vol. 5554, pp. 491–505, Springer, 2009.

[58] T. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing, *International Journal of Human-Computer Studies*, **43**(5-6), pp. 907–928, 1995.

[59] N. Guarino, Formal Ontology and Information Systems, in *Formal Ontologies in Information Systems: Proceedings of the 1st International Conference (FOIS'98)*, pp. 3–15, IOS Press, 1998.

[60] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool Publishers, San Rafael, 2011.

[61] S. Hellmann, J. Unbehauen, A. Zaveri, J. Lehmann, S. Auer, S. Tramp, H. Williams, O. Erling, T. Thibodeau Jr, K. Idehen, A. Blumauer and H. Nagy, Report on Knowledge Extraction from Structured Sources, LOD2 Project, Deliverable 3.1.1, available at: `http://static.lod2.eu/Deliverables/deliverable-3.1.1.pdf`, 2011.

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

[63] M. Hert, G. Reif and H. C. Gall, Updating Relational Data via SPARQL/Update, in F. Daniel, L. Delcambre, F. Fotouhi et al., eds., *Proceedings of the 2010 EDBT/ICDT Workshops*, ACM, 2010.

[64] M. Hert, G. Reif and H. C. Gall, A Comparison of RDB-to-RDF Mapping Languages, in C. Ghidini, A.-C. Ngonga Ngomo, S. Lindstaedt and T. Pellegrini, eds., *Proceedings of the 7th International Conference on Semantic Systems (I-SEMANTICS 2011)*, pp. 25–32, ACM, 2011.

[65] G. Hillairet, F. Bertrand and J. Y. Lafaye, MDE for Publishing Data on the Semantic Web, in F. Silva Parreiras, J. Z. Pan, U. Assmann and J. Henriksson, eds., *Transforming and Weaving Ontologies in Model Driven Engineering: Proceedings of the 1st International Workshop (TWOMDE 2008)*, pp. 32–46, 2008.

[66] W. Hu and Y. Qu, Discovering Simple Mappings between Relational Database Schemas and Ontologies, in K. Aberer, K.-S. Choi, N. Noy et al., eds., *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWCN 2007 + ASWC 2007)*, *Lecture Notes in Computer Science*, vol. 4825, pp. 225–238, Springer, 2007.

[67] P. Johannesson, A Method for Transforming Relational Schemas into Conceptual Schemas, in *Proceedings of the 10th International Conference on Data Engineering (ICDE 1994)*, pp. 190–201, IEEE, 1994.

[68] D. Jurić, M. Banek and Z. Skočir, Uncovering the Deep Web: Transferring Relational Database Content and Metadata to OWL Ontologies, in I. Lovrek, R. J. Howlett and L. C. Jain, eds., *Knowledge-Based Intelligent Information and Engineering Systems: 12th International Conference (KES 2008)*, *Lecture Notes in Computer Science*, vol. 5177, pp. 456–463, Springer, 2008.

[69] Y. Kalfoglou and M. Schorlemmer, Ontology Mapping: the State of the Art, *The Knowledge Engineering Review*, **18**(1), pp. 1–31, 2003.

[70] V. Kashyap, Design and Creation of Ontologies for Environmental Information Retrieval, in *First Agricultural Service Ontology (AOS) Workshop*, 2001.

[71] 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, in F. Frasincar, G.-J. Houben and P. Thiran, eds., *Proceedings of the CAiSE'06 3rd International Workshop on Web Information Systems Modeling (WISM'06)*, pp. 1050–1061, 2006.

[72] 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.

[73] 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.

[74] M. Korotkiy and J. L. Top, From Relational Data to RDFS Models, in N. Koch, P. Fraternali and M. Wirsing, eds., *Web Engineering: 4th International Conference (ICWE 2004)*, *Lecture Notes in Computer Science*, vol. 3140, pp. 430–434, Springer, 2004.

[75] A. Kupfer, S. Eckstein, K. Neumann and B. Mathiak, Handling Changes of Database Schemas and Corresponding Ontologies, in J. F. Roddick, V. R. Benjamins, S. S.-S. Cherfi et al., eds., *Advances in Conceptual Modeling - Theory and Practice: ER 2006 Workshops*, *Lecture Notes in Computer Science*, vol. 4231, pp. 227–236, Springer, 2006.

[76] M. Laclavík, RDB2Onto: Relational Database Data to Ontology Individuals Mapping, in P. Návrat, P. Bartoš, M. Bieliková, L. Hluchý and P. Vojtáš, eds., *Tools for Acquisition, Organisation and Presenting of Information and Knowledge*, pp. 86–89, 2006.

[77] 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.

[78] G. Lausen, Relational Databases in RDF: Keys and Foreign Keys, in V. Christophides, M. Collard and C. Gutierrez, eds., *Semantic Web, Ontologies and Databases: VLDB Workshop (SWDB-ODBIS 2007)*, *Lecture Notes in Computer Science*, vol. 5005, pp. 43–56, Springer, 2007.

[79] G. Lausen, M. Meier and M. Schmidt, SPARQLing Constraints for RDF, in A. Kemper, P. Valduriez, N. Mouaddib et al., eds., *Advances in Database Technology: Proceedings of the 11th International conference on Extending Database Technology (EDBT '08)*, pp. 499–509, ACM, 2008.

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

[81] D. Levshin, Mapping Relational Databases to the Semantic Web with Original Meaning, in D. Karagiannis and Z. Jin, eds., *Knowledge Science, Engineering and Management: Third International Conference (KSEM 2009)*, *Lecture Notes in Computer Science*, vol. 5914, pp. 5–16, Springer, 2009.

[82] M. Li, X. Du and S. Wang, A Semi-Automatic Ontology Acquisition Method for the Semantic Web, in W. Fan, Z. Wu

and J. Yang, eds., *Advances in Web-Age Information Management: 6th International Conference (WAIM 2005)*, *Lecture Notes in Computer Science*, vol. 3739, pp. 209–220, Springer, 2005.

[83] J. Lu, F. Cao, L. Ma, Y. Yu and Y. Pan, An Effective SPARQL Support over Relational Databases, in V. Christophides, M. Collard and C. Gutierrez, eds., *Semantic Web, Ontologies and Databases: VLDB Workshop (SWDB-ODBIS 2007)*, *Lecture Notes in Computer Science*, vol. 5005, pp. 57–76, Springer, 2007.

[84] L. Lubyte and S. Tessaris, Automatic Extraction of Ontologies Wrapping Relational Data Sources, in S. S. Bhowmick, J. Küng and R. Wagner, eds., *Database and Expert Systems Applications: 20th International Conference (DEXA 2009)*, *Lecture Notes in Computer Science*, vol. 5690, pp. 128–142, Springer, 2009.

[85] F. Maali, R. Cyganiak and V. Peristeras, Re-using Cool URIs: Entity Reconciliation Against LOD Hubs, in *Proceedings of the 4th Linked Data on the Web Workshop (LDOW 2011)*, 2011.

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

[87] P. Martin, J. R. Cordy and R. Abu-Hamdeh, Information Capacity Preserving Translations of Relational Schemas Using Structural Transformation, External Technical Report, ISSN 0836-0227-95-392, Ontario, Canada, 1995.

[88] A. Miller and D. McNeil, Revelytix RDB Mapping Language Specification, available at: `http://www.knoodl.com/ui/groups/Mapping_Ontology_Community/wiki/User_Guide/media/RDB_Mapping_Specification_v0.2`, Revelytix, 2010.

[89] B. Motik, On the Properties of Metamodeling in OWL, *Journal of Logic and Computation*, **17**(4), pp. 617–637, 2007.

[90] B. Motik, I. Horrocks and U. Sattler, Bridging the Gap between OWL and Relational Databases, in C. Williamson and M. E. Zurko, eds., *Proceedings of the 16th International Conference on World Wide Web (WWW 2007)*, pp. 807–816, ACM Press, 2007.

[91] V. Mulwad, T. Finin, Z. Syed and A. Joshi, Using Linked Data to Interpret Tables, in O. Hartig, A. Harth and J. Sequeda, eds., *Proceedings of the First International Workshop on Consuming Linked Data (COLD 2010)*, 2010.

[92] I. Myroshnichenko and M. C. Murphy, Mapping ER Schemas to OWL Ontologies, in S.-C. Chen, R. Glasberg, J. Heflin et al., eds., *Proceedings of the 2009 IEEE International Conference on Semantic Computing (ICSC 2009)*, pp. 324–329, IEEE, 2009.

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

[94] P. Papapanagiotou, P. Katsiouli, V. Tsetsos, C. Anagnostopoulos and S. Hadjiefthymiades, RONTO: Relational to Ontology Schema Matching, *AIS SIGSEMIS Bulletin*, **3**(3-4), pp. 32–36, 2006.

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

[96] C. Pérez De Laborda and S. Conrad, Database to Semantic Web Mapping using RDF Query Languages, in D. W. Embley, A. Olivé and S. Ram, eds., *Conceptual Modeling - ER 2006: 25th International Conference on Conceptual Modeling*, *Lecture Notes in Computer Science*, vol. 4215, pp. 241–254, 2006.

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

[98] 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.

[99] A. Poggi, M. Rodriguez-Muro and M. Ruzzi, Ontology-based Database Access with DIG-MASTRO and the OBDA Plugin for Protégé, in K. Clark and P. F. Patel-Schneider, eds., *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*, 2008.

[100] S. Polfliet and R. Ichise, Automated Mapping Generation for Converting Databases into Linked Data, in A. Polleres and H. Chen, eds., *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts*, pp. 173–176, 2010.

[101] 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.

[102] R. Ramakrishnan and J. Gehrke, *Database Management Systems*, McGraw-Hill, New York City, NY, 3rd ed., 2002.

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

[104] S. Ramanujam, V. Khadilkar, L. Khan, M. Kantarcioglu, B. Thuraisingham and S. Seida, Update-Enabled Triplification of Relational Data into Virtual RDF Stores, *International Journal of Semantic Computing*, **4**(4), pp. 423–451, 2010.

[105] 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, *W3C RDB2RDF Incubator Group Report*, 2009.

[106] P. E. Salas, K. K. Breitman, J. F. Viterbo and M. A. Casanova, Interoperability by Design using the StdTrip Tool: an a priori Approach, in A. Paschke, N. Henze and T. Pellegrini, eds., *Proceedings of the 6th International Conference on Semantic Systems (I-SEMANTICS 2010)*, ACM, 2010.

[107] M. Schmidt, T. Hornung, G. Lausen and C. Pinkel, SP^{2}Bench: A SPARQL Performance Benchmark, in J. Li and P. S. Yu, eds., *Proceedings of the 25th International Conference on Data Engineering (ICDE 2009)*, pp. 222–233, IEEE, 2008.

[108] M. Schneider and G. Sutcliffe, Reasoning in the OWL 2 Full Ontology Language using First-Order Automated Theorem Proving, in N. Bjørner and V. Sofronie-Stokkermans, eds., *Automated Deduction - CADE-23: 23rd International Conference on Automated Deduction*, *Lecture Notes in Computer Science*, vol. 6803, pp. 461–475, Springer, 2011.

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

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

[111] J. F. Sequeda, S. H. Tirmizi, O. Corcho and D. P. Miranker,

Direct Mapping SQL Databababases to the Semantic Web: A Survey (Technical Report TR-09-04), University of Texas, Austin, Department of Computer Sciences, 2009.

[112] G. Shen, Z. Huang, X. Zhu and X. Zhao, Research on the Rules of Mapping from Relational Model to OWL, in B. Cuenca-Grau, P. Hitzler, C. Shankey and E. Wallace, eds., *Proceedings of the OWLED'06 Workshop on OWL: Experiences and Directions*, 2006.

[113] A. Sheth and R. Meersman, Amicalola Report: Database and Information Systems Research Challenges and Opportunities in Semantic Web and Enterprises, *ACM SIGMOD Record*, **31**(4), pp. 98–106, 2002.

[114] A. P. Sheth and J. A. Larson, Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, *ACM Computing Surveys*, **22**(3), pp. 183–236, 1990.

[115] E. Sirin and J. Tao, Towards Integrity Constraints in OWL, in R. Hoekstra and P. F. Patel-Schneider, eds., *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, 2009.

[116] K. Sonia and S. Khan, R2O Transformation System: Relation to Ontology Transformation for Scalable Data Integration, in J. Bernardino and B. C. Desai, eds., *Proceedings of the 2008 International Database Engineering & Applications Symposium (IDEAS '08)*, pp. 291–295, ACM, 2008.

[117] L. Stojanovic, N. Stojanovic and R. Volz, Migrating Data-Intensive Web Sites into the Semantic Web, in G. B. Lamont, ed., *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'02)*, pp. 1100–1107, ACM, 2002.

[118] M. Svihla and I. Jelinek, Two Layer Mapping from Database to RDF, in *Proceedinfs of the Sixth International Scientific Conference Electronic Computers and Informatics (ECI 2004)*, pp. 270–275, 2004.

[119] M. Svihla and I. Jelinek, Benchmarking RDF Production Tools, in R. Wagner, N. Revell and G. Pernul, eds., *Database and Expert Systems Applications: 18th International Conference(DEXA 2007)*, *Lecture Notes in Computer Science*, vol. 4653, pp. 700–709, Springer, 2007.

[120] S. H. Tirmizi, J. F. Sequeda and D. P. Miranker, Translating SQL Applications to the Semantic Web, in S. S. Bhowmick, J. Küng and R. Wagner, eds., *Database and Expert Systems Applications: 19th International Conference (DEXA 2008)*, *Lecture Notes in Computer Science*, vol. 5181, pp. 450–464, Springer, 2008.

[121] Q. Trinh, K. Barker and R. Alhajj, RDB2ONT: A Tool for Generating OWL Ontologies From Relational Database Systems, in T. Atmaca, P. Dini, P. Lorenz and J. Neuman de Sousa, eds., *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, IEEE, 2006.

[122] S. R. Upadhyaya and P. S. Kumar, ERONTO: a Tool for Extracting Ontologies from Extended E/R Diagrams, in L. M. Liebrock, ed., *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05)*, pp. 666 – 670, ACM, 2005.

[123] K. N. Vavliakis, T. K. Grollios and P. A. Mitkas, RDOTE - Transforming Relational Databases into Semantic Web Data, in A. Polleres and H. Chen, eds., *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts*, pp. 121–124, 2010.

[124] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov, Discovering and Maintaining Links on the Web of Data, in A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta and K. Thirunarayan, eds., *The Semantic Web - ISWC 2009: 8th International Semantic Web Conference*, *Lecture Notes in Computer Science*, vol. 5823, pp. 650–665, Springer, 2009.

[125] 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.

[126] 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, in A. Gómez-Pérez, M. Gruninger, H. Stuckenschmidt and M. Uschold, eds., *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing*, pp. 108–117, 2001.

[127] Z. Xu, X. Cao, Y. Dong and W. Su, Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies, in H. Dai, R. Srikant and C. Zhang, eds., *Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference (PAKDD 2004)*, *Lecture Notes in Computer Science*, vol. 3056, pp. 464–475, Springer, 2004.

[128] Z. Xu, S. Zhang and Y. Dong, Mapping between Relational Database Schema and OWL Ontology for Deep Annotation, in T. Nishida, Z. Shi, U. Visser et al., eds., *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 548–552, IEEE, 2006.

[129] S. Zhao and E. Chang, From Database to Semantic Web Ontology: An Overview, in R. Meersman, Z. Tari and P. Herrero, eds., *On the Move to Meaningful Internet Systems: OTM 2007 Workshops*, *Lecture Notes in Computer Science*, vol. 4806, pp. 1205–1214, Springer, 2007.

[130] C. Zhou, C. Xu, H. Chen and K. Idehen, Browser-based Semantic Mapping Tool for Linked Data in Semantic Web, in C. Bizer, T. Heath, K. Idehen and T. Berners-Lee, eds., *Proceedings of the WWW 2008 Workshop on Linked Data on the Web (LDOW 2008)*, 2008.