Journal of Web Engineering, Vol. 7, No.1 (2008) 001-024

ONTOLOGY AND DATABASE MAPPING: A SURVEY OF CURRENT IMPLEMENTATIONS AND FUTURE DIRECTIONS

NIKOLAOS KONSTANTINOU DIMITRIOS-EMMANUEL SPANOS NIKOLAS MITROU

National Technical University of Athens

{nkons, dspanos , mitrou}@cn.ntua.gr

Received July 1, 2007
Revised December 1, 2007

In this paper we discuss the problem of mapping relational database contents and ontologies. The motivation lies in the fact that during the latest years, the evolution in Web Technologies rendered the addition of intelligence to the information residing on the Web a necessity. We argue that the addition of formal semantics to the databases that store the majority of information found in the Web is important, in order to make this information searchable, accessible and retrievable. The key technologies towards this direction are the Semantic Web and the ontologies. We analyze in this paper the approaches that have so far been presented in order to exploit the prospects that such collaboration promises. We set the theoretical and practical boundaries of the mapping problem, we delve into the tools that altogether comprise today's state of the art, and we provide a discussion about the benefits and the drawbacks of the existing approaches. We discuss the feasibility and viability of applying the mappings in real world applications as well as the directions that the evolution of current implementations should follow. We conclude by presenting the requirements that should be met in order to provide a more powerful next generation of mapping frameworks.

Keywords: Semantic Web, ontology, database, mapping
Communicated by: D. Lowe & F. Frasincar

1 Introduction

It is generally accepted that the rapid evolution of the Internet has brought up significant changes to information management. The World Wide Web changed the way people create, manage, access, share and retrieve information. The existing data that lie on the Web provide a significant source of information for almost anything imaginable. The research is nowadays focused on how to manage this vast amount of information. Specifically, during the latest years, great effort is being spent towards creating more intelligent services. The Web community is moving towards what is commonly known as Web 2.0. The scope of this effort consists of more efficient information management, systems with faster response, and enhanced intelligence in every aspect of the Web experience. Peer to peer systems, search engines, database technology, the Grid, Web Services and the Semantic Web, all play their part in this new design of the existing infrastructures. One should keep in mind though, that enhancing the Web is not a purpose by itself. The goal is to exploit its capabilities as a Knowledge Base or, in other words, to provide the user with accurate and substantial results in his search for information.

The most powerful tools in users' hands are search engines. Since the World Wide Web became a public global common currency, its content growth made the use of search engines a gateway to information. Without them, the Web in its present form would be useless. Terabytes of data in millions

of Web pages are impossible to be accessed without the use of the current search engine implementations.

There are several reasons that cause this defect of the current form of the Web. The main reason of this weakness is that the largest quantity of existing data on the Web is stored using conventional relational database technology. This information is often referred to as the Deep Web [1], as opposed to the Surface Web comprising all static Web pages. Deep Web pages do not exist until they are generated dynamically in response to a direct user request. As a consequence, traditional search engines cannot retrieve and index Deep Web pages' content.

Moreover, the data that is available on the Web usually abides by ad hoc formalisms. The lack of a common, unified and generally accepted Knowledge Representation formalism impedes data exchange, interoperability and collaboration among Web communities. The institution of a common vocabulary, in combination with the addition of formal semantics is, among others, the goal of the Semantic Web vision. The common knowledge representation formalism is ought to be both human and machine understandable in order to allow inference extraction from existing knowledge.

The creator of the Web, Tim Berners-Lee, proposed the idea of the Semantic Web to overcome the handicaps referenced. As stated in [2], the Semantic Web is about bringing "structure to the meaningful content of Web pages, creating an environment where software agents, roaming from page to page, can readily carry out sophisticated tasks for users". The Semantic Web will not replace the Web as it is known today. Instead, it will be an addition, an upgrade of the existing content in an efficient way that will lead to its integration into a fully exploitable world-wide source of knowledge. In one word, the goal is to bring an order to the chaos nowadays known as the Web or, more precisely, the Deep Web [1].

The key role to this effort is played by ontologies. Their use aims at bridging and integrating multiple and heterogeneous digital content on a semantic level, which is exactly the key idea of the Semantic Web vision. Ontologies provide conceptual domain models, which are understandable to both human beings and machines as a shared conceptualization of a given specific domain [3]. With the use of ontologies, content is made suitable for machine consumption, contrary to the majority of the content found on the Web today, which is primarily intended for human consumption.

Nevertheless, ontologies suffer from a scalability problem. Keeping large amounts of data in a single file is a practice with low efficiency. Ontologies should rather describe content than contain it. Meanwhile, the Semantic Web is not meant to be a candidate technology for Web Engineering, it should instead be an addition to current practices, rather than a substitution. The key in bridging legacy data with formal semantic meaning is the inclusion of mediation between relational database contents and ontologies, which is the exact purpose of the current paper.

The remainder of this paper is structured as follows. In Section 2, we provide a thorough and multi-aspect analysis of the domain of our investigation. In Section 3, we present the tools that altogether comprise today's state of the art in ontology to database mapping[a]. In Section 4, we discuss the results of our survey and we conclude in Section 5, by suggesting a number of ideas that should be taken into account for the design of the next generation tools.

[a] It must be clearly stated that in the current paper, mapping refers to a relation between ontology and database contents and should not be confused with mapping among ontologies

2 Problem Framework

In this Section we provide the definitions and we attempt to approach the mapping problem from several viewpoints. Ontologies and their mapping to relational databases – or vice versa – present theoretical and practical interest when examined by Web engineers, Semantic Web and database theorists and practitioners, and system and knowledge modelers.

2.1 Ontologies in Web Engineering

Software engineers in general have to consider the adoption of a specific working concept in order to efficiently tackle engineering problems that arise in industrial conditions. In order to specify the context that encompasses specific distinct approaches, the term of technical spaces (TS) has been introduced. A TS is a working context with a set of associated concepts, body of language, tools, required skills and possibilities [4]. Before adopting a certain space, Web engineers need to compare the pros and cons of the available approaches. Issues that fall into consideration include expressivity power, ease in interoperability facilities between spaces, standardization level, and a broad list of engineering facilities (i.e. security) most of which are discussed about in the remainder of the current work. Although TSs are difficult to define, they can be easily recognized: XML-based languages (XML TS), Model-Driven Architecture (MDA TS) as defined by the Object Management Group (OMG), Data Base Management Systems (DBMS TS), Ontology TS (OTS) are some of the widespread technical spaces in use. In our work, we take a closer look at DBMS TS and OTS.

The usage of ontologies in systems modeling provides powerful means for the achievement of an abundant system description in description logic (DL) terms. DL allows systems modeling in detail by deriving a concept hierarchy and a corresponding property hierarchy. DL also possesses the unique feature compared to the rest of TSs of defining class membership by solely stating necessary and efficient conditions. However, the strength of the Semantic Web is not restricted in concept description.

Model checking can be realised by the concurrent use of a reasoner, a practice that assures the creation of coherent, consistent models. Model enrichment aims at using ontologies for providing richer descriptions of models from different TSs. The goal is to exploit the ontologies' inference support, the formally defined semantics, the support of rules, and logic programming in general [5]. Hybrid approaches involve models constructed in more than one TS and provide a final model that comprises the mappings between them as well as the models themselves. This approach embraces the current survey scope that is confined to ontology and database mapping. For a further discussion, the interested reader is referred to [6].

2.2 Ontologies and Knowledge Bases

Formally, the term ontology as defined by Gruber in [3] is a "specification of a *conceptualization*" where *conceptualization* is defined in AI as a *structure* <D, **R**> where D is a *domain* and **R** a *set* of relevant relations on D [7]. The set of relations comprises the *intensional* and the *extensional* relations also referred to as *conceptual* and *ordinary* relations respectively. The *domain space* is defined as a *structure* <D, W> where W is a *set* of maximal states of affairs of such domain (also called *possible worlds*). A *conceptualization for* D can be now defined as an ordered triple set **C** = <D, W, **R**>, where **R** is a set of conceptual relations on the domain space <D, W>. In other words, we can describe an

ontology as a set of definitions that associate the names of entities in the universe of discourse with human-readable text describing the meaning of the names and a set of formal axioms that constrain the interpretation and well-formed use of these terms.

More abstractly, an ontology can be defined as a model of a Knowledge Base (KB). Practically, the languages of ontologies are closer in expressive power to First Order Logic (FOL) than languages used to model databases. For this reason, ontologies are considered to be at the *semantic* level, while database schemas lay at the *logical* or *physical* level. The main difference between an ontology and a KB is that the latter offers reasoning as an addition to the model. Thus, it is not possible to extract implicit knowledge from the ontology without the use of reasoning procedures [8]. The term 'ontologies' in the Semantic Web typically refers to two discrete methods of modeling a system's knowledge.

The first one is RDF (Resource Description Framework) [9], in which, the perception of the world is modeled as a graph. The RDF graph is similar to a Directed Labelled Graph, with the difference that RDF allows for more than one edge between nodes [10]. The nodes of an RDF graph are not necessarily connected to each other and it is allowed to find circle paths in the graph. The nodes of an RDF graph contain either resources or literals. The difference between resources and literals is that the latter are not subject to further processing by RDF parsers. Hence, RDF, as indicated by its name, is a language designed to describe resources.

A typical and rather widespread example of a semantic extension of RDF is RDF Schema or RDFS. RDFS does not impose any further syntactic restrictions; it is intended to provide a specific vocabulary – thus restricting the RDF expressivity – in order to be commonly understood. RDFS is the first attempt to bring an order to RDF semantics, e.g. defining the `isSubclassOf` relationship, a relationship missing from RDF. In fact, RDF can be viewed as merely the language, while RDFS as the vocabulary that portrays how RDF can be used to describe Web content. This is the reason why it is usually referred to as RDF(S), a notation that we will also use in the present work.

The second type of ontologies, contains the ones that have their roots in Description Logics (DL) and are usually described in OWL [11, 12], based on standard predicate calculus. Things are more complicated than RDF(S) since the world in OWL is viewed as a set of classes, properties and individuals. According to [13], the OWL vocabulary V can be defined as a set of literals V_L and seven sets of URI references: V_C, V_D, V_I, V_{DP}, V_{IP}, V_{AP} and V_O that denote the class names, the datatype names, the individual names, the data-valued property names, the individual-valued property names, the annotation property names and the ontology names, respectively. The OWL language is the successor of DAML+OIL and is the current recommendation by W3C. The classification support by DL is useful in organizing contexts and context definitions.

DL is a subset of the First Order Predicate Calculus and is not a fixed language; it rather comprises a set of fully defined sublanguages that can be categorized according to their expressive power. The basic DL language is $\mathcal{AL}$ and allows atomic negation, concept intersection, value restrictions and limited existential quantification. In $\mathcal{AL}$, concepts like "persons whose children are all female" can be defined. In order to define more complex concepts, we have to add extra constructors e.g. by adding $\mathcal{N}$ that is number restrictions we get the $\mathcal{ALN}$ language, that can express concepts like "persons who have more than three children". By adding nominals to classes (letter $\mathcal{O}$), allowing the declaration of inverse properties (letter $\mathcal{I}$) and complex concept negation (letter $\mathcal{C}$), setting a hierarchy in concepts (letter $\mathcal{H}$),

the DL that occurs is $\mathcal{ALCHOIN}$. Since $\mathcal{S}$ is shorthand for $\mathcal{ALC}$ with transitive properties, the resulting language containing all these features can be called $\mathcal{SHOIN}$. This language corresponds to the OWL DL[b] language that is widely used in ontologies. This specific subset of DL was chosen because it demonstrates excellent behaviour in algorithms that use an OWL DL ontology to deduce implicit information.

Inference services provided by reasoners (e.g. satisfiablility, subsumption, equivalence, disjointness, consistency) have reached a certain maturity level; they are usually based on tableaux algorithms and are sound and complete. As stated in [8], a Knowledge Base is the combination of an ontology and a reasoner. Although many reasoners exist, the practical choice is made among Pellet [14], FaCT++ [15], KAON2 [16] and Racer [17]. All of them support DIG [18] interoperability which is not a standard yet but it is used by reasoners to exchange HTTP messages with the processes using them.

A KB created using Description Logics consists of two parts. The first part contains all the concept descriptions, the intensional knowledge and is called Terminological Box (TBox). The TBox introduces the terminology and the vocabulary of an application domain. It can be used to assign names to complex descriptions of concepts and roles. The classification of concepts is done in the TBox by determining subconcept/superconcept relationships between the named concepts. It is then possible to form the subsumption hierarchy. The second part contains the real data, the extensional knowledge and is called Assertional Box (ABox). The ABox contains assertions about named individuals in terms of the vocabulary defined in the TBox [19]. A naive approach would be to consider that the TBox of the ontology corresponds to the schema of the relational database and that the ABox corresponds to the schema instance. Unfortunately, things are more complex than that.

2.3 Relational Models

According to [20], a database is a collection of relations with distinct relation names. The relational database consists of a relation schema and a relation instance. The relation schema consists of the schemas for the relations in the database. The relation instance contains the relation instances, whose contents are sets of tuples, also called records. Instances can also be thought of as tables, in which each tuple is a row. All rows have the same number of fields. Fields' domains are essentially the type of each field, in programming language terms (i.e. string, boolean, integer etc.). Relational databases typically, apart from modeling world concepts, also contain additional information such as primary key structures, indexes, stored procedures, triggers and security-related information.

In order to provide access to their contents, databases support many query languages, but SQL is the practical choice. SQL, introduced in [21], is undergoing development for more than fifteen years, is powerful and provides various means of manipulating relational databases. The inputs and outputs of SQL queries are relations. Queries are evaluated using instances of input relations (tables) and produce instances of output relations. SQL is said to be relationally complete – since it is a superset of relational algebra – meaning that every query that can be posed to a relational algebra can be expressed in SQL. In other words, with the use of SQL queries, there is no combination of subsets of tuples in a database that cannot be retrieved.

---

[b] We note that there is an ongoing attempt to extend OWL expressivity by moving from $\mathcal{SHOIN}$ to $\mathcal{SROIQ}$. OWL 1.1 page: http://webont.org/owl/1.1/ accessed on 18-11-2007

The scope of the ontology to database mapping tools is to provide access to the contents of a database through the schema of the ontology. Common approaches, as we will investigate in Section 3, describe mapping mechanisms between ontology classes and database tables. From the Semantic Web point of view, the mappings are capable of corresponding class individuals (alt. instances) to any possible dataset combination, thus significantly extending the storage capability of the ontology. From the view of database theory, the database schema structure is enriched in order to include description logics and answer queries on a higher semantic level.

2.4 Mapping Relational Database Contents to Ontologies

The problem of database-to-ontology mapping is generally regarded as a case of data integration. Things can sometimes be confusing in the specific case of mapping relational database contents to ontology concepts. The difficulties are based on the heterogeneity between these two information storage technologies, since major differences that should be taken into account when considering the problem of mapping exist. For instance, a database schema does not provide explicit and formal semantics for the data, in contrast to ontologies. Moreover, a database schema is not shareable or reusable and it is usually defined for a specific database. On the other hand, an ontology is, by definition, reusable and shareable. Another key difference lies in the development approach. The development of an ontology is an effort that requires coordination among several persons, while a database schema is rarely a result of team-work.

We could claim that databases are similar to Knowledge Bases because of the fact that they are both used to maintain models and data of some domain of discourse [8]. There is, though, a great difference, besides the fact that databases manipulate large and persistent models of relatively simple data while knowledge bases contain fewer but more complex data. Knowledge bases can also provide answers about the model that have not been explicitly stated to it. So, mapping a database to a KB is enhancing the database's ability to provide implicit knowledge by the use of the terminology described in the TBox. In the following Section, we provide a survey of approaches that succeed in mapping the relational database contents to the contents of the ontology. The description language of choice is RDF(S) [9] or OWL-DL [11, 12].

Let us define what a mapping is: according to [22], a mapping is the specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel. A mapping may be expressed as associations, constraints, rules, templates with parameters that must be assigned during the mapping, or other forms yet to be determined. In the present paper, a mapping refers to the creation of a combination between an ontology model and a database model.

The general mapping problem can be considered as a special case of data integration: the problem of schema matching. In the schema matching problem, we are given a pair of two mutually disjoint data sets, the source (local) and the target (global) schema. The idea is to provide a uniform query interface over the data that is mediated. As far as data integration is concerned, there are two approaches. In the Local-As-View (LAV) setting, the source database is modelled as a set of views on the global schema. As local, we refer to local sources or databases and as global we refer to the result of the mediated schema. In the Global-As-View (GAV), the global schema is modeled as a set of

views over the source schema. The major drawback of the GAV approach is that it is necessary to redefine the view of the global schema every time a new source is integrated.

Theoretically, a Data Integration system is a triple <*G, S, M*> where *G* is the global schema, *S* is the source schema and *M* is a set of assertions that relate the elements of the source schema to the elements of the global schema. Generally speaking, the goal of a data integration system is to provide a common interface to various data sources, so as to enable users to focus on specifying what they want. In our case, the source schema is the schema of the relational database, and the target schema is the ontology model. Therefore, the general problem of data integration is reduced to the problem of creating correspondences between sets of relational and ontological data. In order to define the general problem of database and ontology mapping, we assume an environment where we are given:

1. An ontology, expressed in a language such as RDF(S) or OWL. The ontology contents can be viewed as a graph or as a set of triples.

2. A relational database instance, whose contents are stored into tuples.

The general objective is to find mappings and create a set of correspondences relating ontological data – predicates in the ontology – and subsets of the relational data – tuples. The idea is illustrated in the Figure below.

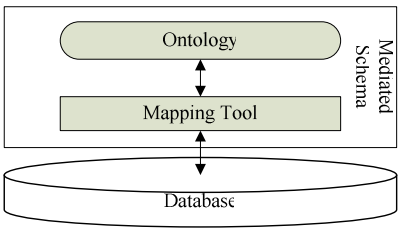

Fig. 1. The problem of ontology and database mediation can be abstractly depicted as the intermediation of a mapping tool between the ontology and the database.

However, one can easily observe that the SQL DDLs (Data Definition Languages) are more expressive than DL languages in some aspects, while the relational databases have a variety of features not included in DL implementations. This heterogeneity is covered in the following points:

- The relational schema may contain further constraints: `DEFAULT`, `NOT NULL`, `UNIQUE`, `AUTO INCREMENT`.
- The attributes of the relational schema can be assigned using rules that define automatic behaviours (triggers): `ON INSERT, ON UPDATE`, `ON DELETE`.
- The majority of the current database implementations incorporate the ability of stored procedures and fragments of code that can be executed with the use of SQL.
- Transactions (`START TRANSACTION`, `COMMIT`, `ROLLBACK`) are widely used in commercial applications and in general where security is crucial.
- Security-related issues in storing and accessing content are still missing from ontologies.

All the previously mentioned constructs cannot be mapped to ontologies. We should not forget that when the mapping process is completed, the goal is to impose queries to the result of the mapping while preserving the initial expressivity.

We noted earlier that, in a simplified point of view, the schema of a relational database can be compared to the TBox of a KB and the schema instance to the ABox. However, there are significant differences between Knowledge Bases and databases, the most profound one being the richer expressiveness offered by the former. The only relationship that can be expressed in the relational model is an IS-A relationship, while the ontological scheme allows for more complex relationships to be stated. In fact, the relational model does not provide enough features that could be used to assert complex relationships among data.

Yet another significant difference is the kind of semantics that holds for each schema. The relational database schema abides by the so called 'closed world assumption'. The system's awareness of the world is restricted to the facts that have been explicitly stated to it. Everything that has not been stated as true fact is false. In a 'closed world', a null value about a subject's property denotes the non-existence i.e. a `NULL` value in the `isCapital` field of a table `Cities` claims that the city is not a capital. The database answers with certainty because, according to the closed world assumption, what is not currently known to be true is false. Thus, a query like 'select cities that are capitals' will not return a city with a null value at a supposed boolean `isCapital` field[c].

On the other hand, a query on a KB can return three types of answers: `true`, `false` and `cannot tell`. The open world assumption states that lack of knowledge does not imply falsity. In this case, information that is not explicitly declared as `true` is not necessarily `false`, it can also be `unknown`. So a question 'Is Athens a capital city?' in an appropriate schema will return 'cannot tell' if the schema is not informed while a database schema would clearly state `false`, in the case of a null `isCapital` value.

2.5 Querying the result: SPARQL vs. SQL

It can be argued that it is unfair to compare the two languages, yet. On one hand, SPARQL is still under design as far as it concerns aspects like its formal semantics, while SQL has been present for many years now. SQL is currently a mature, widely used standard, whilst SPARQL still is in its infancy and has to evolve significantly to reach SQL's capabilities. SPARQL [23], formerly known an BrQL, is an extension of RDQL and is currently a candidate recommendation for standardizing in the W3C. The popular Semantic Web framework Jena uses ARQ, an implementation[d] of the SPARQL query language. We have to mention that in SPARQL, the graph level is not taken into account as in other query languages, (e.g. RQL [25], available in the Sesame system [26]), but instead the data is modeled as a set of triples. Apart from this feature, SPARQL bears a lot of similarities to SQL, like the `SELECT FROM` syntax; however, although it was designed to have an SQL-like syntax, in its base, it can be regarded as simple triple pattern statements. This is because of the nature of the underlying data that is typically stored in triples. The differences concerning the selection of data can be summed as follows:

---

[c] The example is inspired by the one found in http://en.wikipedia.org/wiki/Open_World_Assumption, accessed on 19-11-2007

[d] Since query languages for OWL are not mature yet [24], SPARQL is nowadays the most promising solution for querying ontology data. A list with SPARQL implementations can be found in http://esw.w3.org/topic/SparqlImplementations, accessed on 19-11-2007

- In contrast to SQL, SPARQL does not support nested queries.
- The `SELECT WHERE LIKE` statement is missing, a practice that is typically implemented in keyword-based queries.
- SPARQL does not implement a similar construct to `GROUP BY` offered by SQL.
- Aggregation is also missing, in the form of `MIN`, `MAX`, `SUM`, `COUNT` or `AVG` functions.

Nevertheless, SPARQL embodies a variety of interesting features not present in SQL. A feature that can be met in almost all of the query languages for RDF is the use of the `OPTIONAL` operator that does not modify the results in case of non-existence. Moreover, the `CONSTRUCT` instruction reassures that the results will be expressed as an RDF graph and also allows the building of new graphs based on the result sets. Finally, SPARQL can extract resource descriptions, using `DESCRIBE`. This property is not fully defined yet but it is designed for future use. In [27], an approach of how to model simple SQL queries to SPARQL queries is presented.

3 Current Practice and Research

Due to the interdisciplinary nature of the work, namely ontology and database integration, the related work is far and wide. Therefore, we have to state explicitly that in our research we did not take into account tools or frameworks that allow simple storage of ontology data in a relational database. Similar approaches, like 3Store [28], Corese [29], Sesame [26], Kowari [30] or Instance Store [31] are out of the scope of the current survey. These tools have in common that the database is used to store ontological data but the user does not define mappings; he can only interact with the ontology data layer.

MOMIS [32, 33] deals with the heterogeneous information sources integration problem. It is a pool of tools, providing an integrated access to heterogeneous information stored in traditional databases or file systems, as well as in semistructured data sources. The system uses ODL_{I3}, an object-oriented language derived from the standard ODMG (Object Database Management Group), in order to represent the semantics underlying the schema. The main components of MOMIS architecture are wrappers managing all local information sources, a mediator comprising a global ontology builder and a query manager, a tool - called ARTEMIS - performing classification of local ODL_{I3} classes for the synthesis of global ODL_{I3} classes, and another tool - called ODB-Tools Engine - based on the OLCD Description Logics, which infers new relationships between local ODL_{I3} classes and contributes to the generation of a common Thesaurus. The system's wrappers are placed on top of the information sources and are responsible for translating the schema of the source into the ODL_{I3} language. They also perform the translation of a query expressed in the OQL_{I3} language into a local request executable by the query processor of the corresponding source. The translation of the source schema into the ODL_{I3} language is based on the, rather elementary, rules: i) a relation name (e.g. a database table) corresponds to an ODL_{I3} class, and ii) for each relation attribute, an attribute is defined in the corresponding ODL_{I3} class.

Clio [34, 35] is a tool that infers mappings from one set of relational tables and/or XML data to another, but, with minor changes, it could be applied to an ontology schema as well. At the core of the system are the mapping generation component and the query generation component. The mapping generation component takes as input correspondences between the source and the target schemas and

generates a schema mapping consisting of a set of logical mappings (declarative assertions, in fact) that provide an interpretation of the given correspondences. The query generation component has then the role to convert a set of logical mappings into an executable transformation query. Query generation consists of a generic module, independent of a particular execution language, and of a number of pluggable components that are specific to each execution language: SQL, SQL/XML, XQuery and XSLT. At any time during the design, the user can view, add and remove correspondences between the schemas through a GUI component and can inspect and edit the generated logical mappings. Clio runs a data chase algorithm – introduced in [36] – for the generation of mappings, which are then represented with the use of an internal notation-mapping language that includes Skolem functions as well.

One of the few approaches using reverse engineering is the one proposed by Stojanovic et al. [37], mapping a given relational schema into an existing ontological structure. They first capture information from a relational schema through reverse engineering and, by using a set of mapping rules, they analyze the obtained information in order to map database entities to ontological entities. Then, they perform evaluation, validation and refinement of the mapping, checking whether all relational entities are mapped to corresponding ontological entities and finally, data migration is performed, involving the creation of ontological instances based on the tuples of the relational database. **KAON-REVERSE**, a tool for semi-automatically connecting relational database to ontologies[e], has been used in the implementation of this approach, for the automation of the mapping process.

D2R MAP was first presented in [38] and provides means to declaratively state ontology-to-database mappings. D2R MAP is based on XML syntax and constitutes a language provided to assign ontology concepts to database sets. It allows mappings of complex relational structures to OWL/RDFS ontologies, by employing SQL statements in the mapping rules, without having to change the existing database schema. More precisely, a mapping is represented by a `ClassMap` element, containing attributes such as `SQL` (the SQL statement describing the data set), `groupBy` and `uriPattern` attributes for the creation of ontological instances. Results are extracted in RDF, N3, RDF or Jena [39, 40] models. D2R MAP can handle highly normalized table structures, where instance data is spread over several tables. However, it fails to map low structured databases because of its limited expressiveness.

D2RQ [41] builds on the above concept of D2R MAP, thus retaining the `ClassMap` element, but with a slightly different syntax. D2RQ[f] is implemented as a Jena graph, wrapping one or more local relational databases into a virtual, read-only RDF graph. With the use of D2RQ, an application can query a non-RDF database using RDQL. D2RQ rewrites RDQL queries and Jena API calls into database-specific SQL queries. The result sets of these SQL queries are transformed into RDF triples which are passed up to the higher layers of the Jena framework. Since the corresponding mappings between the database and RDF are created manually, they have to be rechecked after each evolution of the schema. Especially in environments with constantly changing schemas, this is not easily manageable.

[e] http://kaon.semanticweb.org/alphaworld/reverse/view, accessed on 18-11-2007
[f] http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/, accessed on 18-11-2007

RDF Gateway[g] is commercial software having similar functionalities. It connects legacy database resources to the Semantic Web via its SQL Data Service Interface. The SQL Data Service translates an RDF based query (expressed in RDFQL, a query language the developers have come up with) to an SQL query and returns the results as RDF data. However, it uses a native database engine that stores data to a single table, according to the vertical table approach, a choice that casts serious doubts on the system's performance, since in this case queries have to search the whole database in order to return an answer.

eD2R (extended D2R) [42] is an extension of D2R MAP, adding operation and condition elements expressed in terms of elemental functions allowing the definition of complex and conditional transformations on field values and is based on techniques, such as keyword search or natural language processing. **R_2O** (Relational to Ontology) [43] is another extensible, declarative, XML-based language to describe, expressively enough, mappings between relational DB schemas and ontologies implemented in RDF or OWL. Like D2R MAP, R_2O allows the definition of explicit correspondences between components of two models. R_2O is an RDBMS independent high level language that works with any DB implementing the SQL standard. An R_2O mapping defines how to create instances in the ontology in terms of the data stored in the database. The approach suggested by the authors consists of creating a mapping description document using R_2O with all the correspondences between the components of the DB's SQL schema and those of the ontology. Such mappings are then processed automatically by a mapping processor to populate the ontology.

The Ontomat Application Framework, introduced in [44] was one of the first prototypes for database and semantics integration. It was a front-end tool built upon a component-based architecture. **Ontomat Reverse** [45] is part of the framework and offers the means to semi-automatically realize mappings (via a set of mapping rules) between ontology concepts and databases through JDBC connections. Nevertheless, it only supports RDF graphs and mappings between database tables on the server and ontology classes on the client.

Considering the problem of deep annotation in the Semantic Web, Volz et al. [45] describe a framework of metadata creation where Web pages are generated from a database. They consider two ways for the deep annotation of the database; directly by annotation of the logical database schema or indirectly by annotation of the Web presentation generated from the database contents. We are particularly interested in the former case, where the database schema is being mapped into a given ontology. For the automation of the mapping process, they use Ontomat Reverse, which automatizes some phases in that mapping process, particularly capturing information from the relational schema, validation of the mapping process and data migration. More precisely, database relations are mapped to ontology concepts if a lexical agreement in the naming exists (using edit distance as a measure), attributes are mapped to corresponding datatype properties, if such properties are defined for the concept or one of its sub-concepts, and associations between database relations, expressed via foreign keys, are mapped to object properties. Nonetheless, the user may refine or remove automatically generated mappings or even create new ones.

A recent solution to the heterogeneous database integration problem has been proposed by Dou and LePendu [46] through their **OntoGrate** architecture, transforming relational schema into

[g] http://www.intellidimension.com, accessed on 18-11-2007

ontological representation, while allowing users define the mappings at the ontological level using bridge-axioms. The OntoGrate approach involves ontology-based schema representation, first order logic (FOL) inference, and some SQL wrappers. To model database schemas, concepts and the relationships (mappings) among them, the authors use the Web-PDDL ontology language, a FOL language. The mappings have the form of bridging axioms expressed in Web-PDDL. The SQL wrappers lie above the databases and translate FOL queries to SQL queries (by a direct application of transformation rules), which are then executed on the appropriate SQL database using JDBC. The SQL wrappers also translate the resulting SQL record sets to FOL assertions.

In order to tackle the problem of mediation, in [47] Dou et al. present PDDSQL, a language that automatically translates between SQL and Web-PDDL, a FOL language with a Lisp-like syntax that expresses ontologies, data instances (facts), queries and mapping rules between different ontologies. The same team has also developed a syntax translator called PDDOWL that can translate OWL-QL [48] to Web-PDDL as well.

Another approach worth mentioning is that of de Laborda and Conrad [27], who, seeking a way to make the Semantic Web vision more of a reality, examine whether the combination of **Relational.OWL** as a Semantic Web representation of relational databases and a semantic query language like SPARQL [23] could suffice. Relational.OWL is a technique to extract the semantics of a relational database and transform it into an RDF/OWL ontology. The data items are represented as instances of this data source specific ontology. Relational.OWL, using the techniques provided by the Web Ontology Language OWL, defines classes like `Table` or `Column` and specifies possible relationships among these classes. Thus, an automatic transformation mechanism is created, from data stored in relational databases into a representation which can be processed by virtually any Semantic Web application. The shortcoming of this approach is that all the relational data have to be stored into RDF/OWL format before querying, which would be impractical if the RDBMS contains a huge volume of data.

MAPONTO [49] is a semi-automatic tool that assists users to discover plausible semantic relationships between a database schema and an ontology, expressing them as logical formulas. The tool - implemented as a tab plugin for the popular ontology editor Protégé[h] - expects the user to provide simple correspondences between atomic elements used in the database schema and those in the ontology, in order to generate a list of candidate rules for each individual component in the database schema. The main idea underlying MAPONTO[i] is to represent the ontology as a graph consisting of nodes (concepts) connected by edges (properties). Then, the tool finds the minimum spanning tree (Steiner tree) connecting the concepts having datatype properties corresponding to table columns, and encodes the tree into a logical formula by joining the concepts and properties encountered in it. However, the authors do not mention how they deal with schema evolution in the original database. Apparently, the corresponding mappings would have to be updated manually.

Next, we present a comparative overview of the most important approaches mentioned above. In Table 1, we specify the ontology language, the RDBMS and the semantic query language supported for each tool. Moreover, the degree of automation in the mapping process offered by each tool is stated. In Table 2, we state the methodology and techniques followed by each approach, as well as the

[h] http://protege.stanford.edu/, accessed on 2-11-2007
[i] http://www.cs.toronto.edu/semanticweb/maponto/, accessed on 18-11-2007

components of the models (ontological and database) that each tool allows to be mapped. The capability of consistency checks and the interaction of each tool with the user are also included among the criteria.

Table 1. Classification of the approaches.

Tool	**Ontology Language**	**RDBMS**	**Semantic Query Language**	**Information needed/Degree of automation**
D2RQ	RDF, DAML+OIL	Any RDBMS offering JDBC or ODBC access	RDQL	Both manual and automatic
D2R MAP	RDF	Any RDBMS offering JDBC or ODBC access	None	Manual
Clio	N/A	Any	N/A	Semi-automatic
MOMIS	ODL_{I3}	Any	OQL_{I3}	Semi-automatic
R_2O	RDF/OWL	Any SQL-implementing RDBMS	None	Manual
OntoGrate	Web-PDDL	Any SQL-implementing RDBMS	Web-PDDL	Manual
MAPONTO	OWL	Any SQL-implementing RDBMS	None	Semi-automatic
Relational.OWL	RDF/OWL	DB2, MySQL, Oracle	Any language that can query an OWL ontology	Automatic

Table 2. Classification of the approaches (continued).

Tool	**Methodology-Techniques**	**Components mapped**	**Consistency Checks**	**User Interaction**
D2RQ	Language for mappings description	DB tables, columns, primary/foreign keys	Yes, through the Jena API	No graphical interface, user provides mappings in the form of a proprietary language
D2R MAP	XML-based language	DB tables, columns, keys	No	No graphical interface, user provides mappings in the form of a mapping language
Clio	Data chase algorithm, attribute matching algorithm. Mappings are described in SQL, XQuery or XSLT	DB tables, columns, keys	No	GUI, enabling the user to remove, add or edit mappings. Moreover, the user must provide correspondences between target and source schema.
MOMIS	Affinity calculus, clustering techniques, use of WordNet	DB tables and columns	Yes, via ODB-Tools Engine	Initial annotation of local sources' schemata, provided by user through a GUI. User can intervene at any point of the procedure, by adding or removing relationships between schema elements.
R_2O	Ontology populated with instances according to a set of mappings specified by the user	DB tables, columns, foreign keys	No	No graphical interface, user provides mappings in the form of a proprietary language
OntoGrate	Mappings described as bridging axioms	DB tables, columns, integrity constraints, keys	Yes, via the OntoEngine reasoner	Query interface

MAPONTO	Shortest path finding between concepts of the ontology	DB tables and columns	No	The user should provide correspondences between database and ontology
Relational.OWL	Creation of one class per database component mapped	DB tables, columns, primary/foreign keys, datatypes	No, ontology is described in OWL Full	None

As it can be seen from the above tables, there is a wide spectrum of approaches in the database and ontology collaboration issue, each one using an arbitrary methodology and various techniques. This just showcases the evolving nature of this field, emphasized by the lack of a common procedure. In order to summarize in a more concise way these approaches, we have formed a simple taxonomy for their classification, shown in Figure 2. We should keep in mind, however, that each taxonomy class is quite heterogeneous, containing tools that differ substantially from each other in certain aspects.

The whole set of the approaches in the database and ontology collaboration field can be classified into two broad categories, according to whether the user already has in his possession an ontology to use. In the first case, where there is a given ontology model that needs to be mapped to a relational database schema, approaches such as KAON-Reverse, R_2O, Ontomat Reverse and MAPONTO can be included. This category of tools can be subdivided itself in two other classes of approaches, one defining mappings manually - that would be the case of R_2O, in which the user states explicitly mappings between the relational schema and the ontology model - and another following a semi-automatic mapping procedure, as is the case with the rest of the aforementioned tools that are capable of detecting some correspondences, using heuristics, measures of lexical proximity and other techniques.

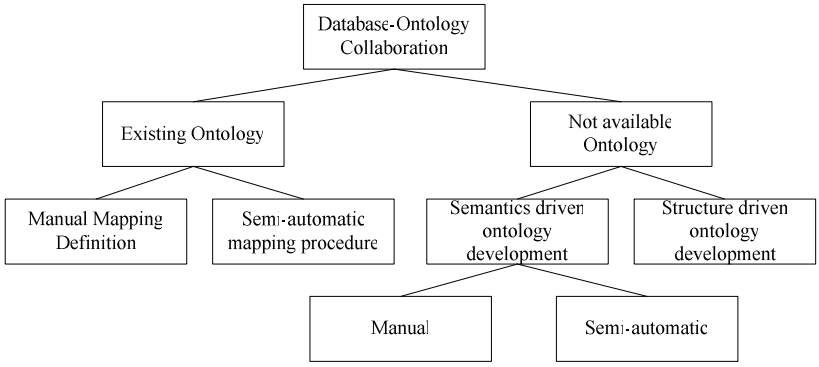

Fig. 2. A simple taxonomy of mentioned approaches.

The second group of approaches does not encompass a given ontology, but instead builds an ontology, based on the given relational schema. Tools in this group include, among others, MOMIS, D2RQ, OntoGrate and Relational.OWL. Some of these tools use the generated ontology model merely as an integration medium between two or more relational schemata, while for the rest of them, the generation of an ontology is the ultimate goal, enabling the addition of semantics to data stored in relational databases. Whatever the rationale of the whole application is, we can discern two cases of ontology model building: semantics driven and structure driven ontology development. Under the latter category, we can only include Relational.OWL, which uses predefined ontology classes and

properties (e.g. Table, Column, hasColumn) in order to describe just the structure of a relational database, hence ignoring the semantics of the relational model. On the contrary, semantics driven ontology development is based on the detection of concepts and relationships between them in the relational schema and the construction of corresponding ontology classes and properties. This construction can either be stated explicitly, thus leading to a manual ontology development - that is the case with D2RQ - or it can be carried out semi-automatically, as in MOMIS or OntoGrate, where the system proposes certain correspondences and the user can accept or reject them or even add new ones.

3.1 Proof of concept

To give an even clearer picture of some of the above tools, we have built a MySQL database called 'travel' and mapped some of its contents to an ontology that contains travel information[j], using the tools that were available for public use. Our custom database schema consists of, among others, the tables 'cities', 'hotels', 'museums' and 'activities' and is populated with data that, plausibly, are semantically linked to the domain of the travel ontology. The table 'activities' contains information about activities offered to tourists, including their description and type (e.g. surfing, hiking), while the contents of the rest of the tables can be easily understood, as their names are self-explanatory. This example is based on the one illustrated in [50], describing yet another approach in the database-ontology collaboration problem, which, alas, was deemed quite novice to mention it among the other, more complete approaches.

D2RQ, as mentioned before, is a tool that, in fact, takes as input a database and a document containing some mappings and produces as output an ontology. An interesting feature of D2RQ is the automation of the generation of a mapping document, if desired by the user. Due to space limitations, we are here presenting only the part of the mapping document that refers to table 'cities', which has three columns, namely 'id', which is its primary key, 'name' and 'isCapital', with a null default value. A segment of the mapping of the table 'cities' is expressed in D2RQ, as follows, in N3 notation:

```
map:cities a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "cities/@@cities.id@@";
    d2rq:class vocab:cities; .
map:cities__label a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:cities;
    d2rq:property rdfs:label;
    d2rq:pattern "cities #@@cities.id@@"; .
map:cities_id a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:cities;
    d2rq:property vocab:cities_id;
    d2rq:column "cities.id";
    d2rq:datatype xsd:int; .
```

We can observe that an ontology class named 'cities' is created and a datatype property, 'cities_id' – accordingly for the columns 'cities_name' and 'cities_isCapital', whose mappings are not shown here – corresponding to the columns of the database table 'cities'. Moreover, the data stored in the database are used to create ontological instances and that is why URI patterns are specified in the mapping specification above. An instance of the class 'cities' is presented next, again in N3 notation:

```
<http://localhost/cities/6> a vocab:cities ;
       rdfs:label "cities #6" ;
```

[j] The travel ontology can be found at: http://protege.cim3.net/file/pub/ontologies/travel/travel.owl.

```
    vocab:cities_id "6"^^xsd:int ;
    vocab:cities_name "Patras" .
```

Relational.OWL resembles D2RQ in that it does not require as input an ontology but instead builds one, according to some predefined rules with the use of standard classes and properties. Relational.OWL does not examine the semantics of the database; it only considers its structure, just like D2RQ, when run in automatic mode. Once more, we present only the segment of the generated RDF ontology which refers to table ‘cities’:

```
<rdf:Description rdf:ID="Cities">
  <rdf:type rdf:resource="&dbs;Table"/>
  <dbs:isIdentifiedBy rdf:resource="#PK0"/>
  <dbs:hasColumn rdf:resource="#Cities.id"/>
  <dbs:hasColumn rdf:resource="#Cities.name"/>
  <dbs:hasColumn rdf:resource="#Cities.isCapital"/>
</rdf:Description>
<rdf:Description rdf:ID="Cities.id">
  <rdf:type rdf:resource="&dbs;Column"/>
  <rdfs:domain rdf:resource="#Cities"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
  <dbs:length>6</dbs:length>
</rdf:Description>
<rdf:Description rdf:ID="Cities.name">
  <rdf:type rdf:resource="&dbs;Column"/>
  <rdfs:domain rdf:resource="#Cities"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>
<rdf:Description rdf:ID="Cities.isCapital">
  <rdf:type rdf:resource="&dbs;Column"/>
  <rdfs:domain rdf:resource="#Cities"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
  <dbs:length>3</dbs:length>
</rdf:Description>
<rdf:Description rdf:about="PK0">
  <rdf:type rdf:resource="&dbs;PrimaryKey"/>
  <dbs:hasColumn rdf:resource="#Cities.id"/>
</rdf:Description>
```

The Relational.OWL output bears similarities to the D2RQ output in the number of classes and properties created; to be more precise, one class per table and one property per column. In addition to this, one class describing the primary key of each table is created as well one class for every foreign key in the database, the latter case not being shown here. All these classes are predefined in Relational.OWL, thus allowing it to generate an ontology describing the input database schema no matter how complicated it may be. Another ontology, linked with the “schema” ontology, is used for the storage of the instances created, in order to contain all database data. An instance of table ‘cities’ follows, where the namespace j.0 refers to the “schema” ontology file:

```
<j.0:Cities>
  <j.0:Cities.id>6</j.0:Cities.id>
  <j.0:Cities.name>Patras</j.0:Cities.name>
</j.0:Cities>
```

R_2O differs from both previous approaches, as it constructs mappings between a database and an existing ontology. As such a language, it requires description of the database schema in addition to the definition of the mappings, thus imposing an additional description overhead for the user. In our example, we are trying to map the database table ‘cities’ to the ontology class ‘City’ of the travel ontology we mentioned earlier. Moreover, we specify the desired URI of the instances that will

populate the ontology. The appropriate database schema description and mapping definition in R_2O would look like this:

```
dbschema-desc
  name Travel
  has-table
    name Cities
    keycol-desc
      name id
      columnType integer
    nonkeycol-desc
      name name
      columnType string
    nonkeycol-desc
      name isCapital
      columnType integer
```

Database Schema description in R_2O

```
ontology http://www.owl-
ontologies.com/travel.owl#
conceptmap-def
  name City
  identified-by Cities.id
  uri-as
    concat
      arg-restriction
        on-param string1
        has-value string "City_"
      arg-restriction
        on-param string2
     has-column Cities.name
```

Mapping Definition in R_2O

4 Discussion

Our research indicates that database to ontology mapping is still a very active domain. Prevalent Web trends like blogging, RSS, and the FOAF network provide the Web with new, constantly updated material that is often accessed. Furthermore, the emergence of the Semantic Web, made it possible to publish and access far more ontologies than knowledge engineers ever thought that it would be possible to build [51]. The community of Web Engineering has created so far numerous ontologies[k] that make the task of handling them as important as developing them. Nevertheless, the current state of the art in mapping tools is still in its infancy.

First of all, *automation*, an issue of utmost importance, is not adequately supported. Among the tools we investigated, only D2RQ [41] and Relational.OWL [27] offered automated mappings. Also, we did not see a programmatic API that would enable Web Engineers to author their custom mappings in order to construct a seamless mapping layer – with the exception of D2RQ [52]. Moreover, no middleware software exists that will offer transparent transaction between the two levels of information.

Another important absence is the *support for collaborative authoring* of the mapping instructions. The tools investigated were all lacking team editing support. It is our conviction that the complicated nature of the mapping problem in addition to the distributed architecture of Web systems underlines the necessity of team work. Especially when each contributor is an expert on a specific domain, the need for a collaboration platform is even greater.

Relational databases are widely used in real-world scenarios where *security* is an important issue. Practice has affected SQL evolution so as to make it completely safe concerning both user permissions and actual data storage. Security is always a headache for Web Engineers because exposure to the public requires substantial security mechanisms. It would be very eligible to see an approach taking into consideration security-related issues. In our opinion, security should be faced in two levels. First, the lack of security entailed by the storage of the ontology is a matter that has to be dealt with and second, we would like to see something similar to the connection string, commonly used in the database world. D2RQ was the only approach that tried to consider this security issue, by enabling

[k] Until today, Swoogle [57] has indexed more than 10,000 ontologies. For more detailed and up-to-date statistics visit http://swoogle.umbc.edu/index.php?option=com_swoogle_stats, accessed on 18-11-2007

conditional mappings. Consequently, only data that satisfy certain conditions can be accessible through the ontology, when a conditional mapping is applied, thus allowing the user to preserve the confidentiality of his data.

Among our observations was that database-and-ontology collaboration was preferred to database-to-ontology migration. All the tools we reviewed fall to the first category, with the exception of Relational.OWL [27] that proposes a direct manipulation of the relational data to the RDF/OWL format and then the process by a reasoner. In the rest of the tools, the aim is to integrate the data with the use of an ontological mapping tool.

The Semantic Web can be regarded as an open and distributed system. Heterogeneity cannot be avoided in such systems. According to [53], heterogeneity can be met in four levels; syntactic, terminological, conceptual, semiotic/pragmatic. Several surveys have been provided in the field of ontology collaboration, such as [31, 55]. The vision is to create a Web that offers a great deal of human-centric services, in contrast to the current monolithic approaches. It should be taken into account that interoperability among the various mapping methodologies is an important task. Methods and tools should be provided in order to allow interoperability at an even higher level of information integration. Then again, the mapped data should be connected to each other in order to implement the Semantic Web ultimate goal.

4.1 Benefits and Drawbacks

In order for the ontology and database mediation to satisfy real world needs, one has to take into account numerous factors. The first thing is to balance between the resources spent and the benefits gained. A company that considers a possible adoption of similar technologies would weigh the benefits and drawbacks of the proposed integration tool, performing a typical, crucial in decision making, cost-benefit analysis.

Among the obvious benefits is that querying the system will be more robust, since it will incorporate Semantic Web technologies. However, in order to achieve this goal, we should be aware that a single ontology by itself cannot usually cover the needs of a medium or large-scale project and it cannot be flexible in implementation, either. Detailed descriptions of the conceptualisation of the world usually span over various domains and require technical expertise in many fields. A common goal is the unification of Knowledge Representation, in the form of the institution of a common vocabulary. The key technologies in the Web Engineering are databases and ontologies. According to [1], the majority of the data that lies in the Web is stored in some form of a relational database. We assume that the relational schema and the ontology were developed independently, an assumption which complicates things. There are usually some parts of the database which are not included in the ontology and vice versa. The majority of the current applications is developed ad hoc, thus impeding interoperability, integration and collaboration among various systems.

A representative example of mediocre behaviour is the infamous OpenCyc ontology [56]. The project officially began on 1984 by Doug Lenat and is under development until today. The OWL version of OpenCyc is a huge file larger that occupies more than 700 MB of space and takes approximately 9 hours to load into Protégé[l]. Using Swoogle [57], we found reduced versions of

[l] www.opencyc.org, accessed on 18-11-2007

OpenCyc but this does not solve the problem. OpenCyc has been connected to terms of WordNet [58], the open dictionary developed mainly in the Princeton University. Due to its large size, OpenCyc is difficult to handle. One can easily map concepts of a custom ontology to concepts in OpenCyc but it is difficult to maintain, develop, use and share ontological descriptions of this size. It would be preferable to segregate the large ontology into smaller fragments, in order for a community to work parallel on a common task.

Drawbacks include concerns such as the viability of the whole integrated system, an issue that arises after the creation of the mapping. It cannot be disputed that in real-world scenarios, the manual establishment of mappings between database schemas and ontologies is considered as an additional burden in the development process. It is a time-consuming and error-prone task. Moreover, when the responsible for the mapping process will have to declare a non-trivial mapping, he will often need domain expertise in both the ontological and the relational models. In addition, ontologies and databases are subject to changes that can easily cause the mappings to be obsolete or, worse, erroneous. Therefore, without some maintenance, a static mapping will not satisfy real-world needs. Moreover, it should be taken into account that usually, companies are unwilling to provide mechanisms to access their data. In other words, questions that have to be answered include who, how, and how often will he or she realise the mappings?

4.2 Soundness, Completeness and Performance

Among the issues of major importance in mapping the Web Data is to consider the practical value of the result. Information theorists require the result to be sound and complete. A system is sound when every answer it returns is valid and complete when it can return answers about the whole extent of its knowledge. Nevertheless, as far as practical issues are concerned, it is well known that even reasoning with OWL Lite can be of high worst-case complexity [59]. Therefore, while designing, implementing or using such a system, one should be aware of its limitations. The current systems have to balance between soundness, completeness and performance.

Generally, results returned to the user are expected to be returned in less than a second. Users would rather prefer incomplete results in reasonable time – less than one second – than thorough results but having to wait for them. In [47] it is argued that applying a sound but incomplete approach seems to be a convincing direction.

As far as it concerns scalability measurements in the Semantic Web, the standard benchmark solution is the Lehigh University Benchmark (LUBM) [60] that has taken a lot of criticism [59] but remains nevertheless the practical solution. In [59], the authors claim that, in order to meet real-world needs, a reasoning system has to offer a sufficiently expressive query language as well as a flexible and efficient communication interface. LUBM suggests an incomplete query answering procedure and makes use of information about individuals of a classified TBox, while UOBM [61] extends the LUBM in terms of expressivity. This is accomplished by adding extra axioms in the TBox using the full expressivity of OWL Lite (UOBM Lite) and OWL DL (UOBM DL). Unfortunately, we did not run into any benchmarking measurements for any of the tools presented here, with the exception of D2RQ, whose results are presented in [52].

5 Concluding Remarks

Based on the discussion in this paper and inspired from [62], where Uren et al. set the requirements for content annotation, we set the requirements for the ontology to database mapping that should be considered when designing the next generation tools. The collaboration between the two different technologies, in order to be widespread has to compete with existing methodologies, such as Model Driven Engineering (MDE), that already tackle the following issues. The necessity of clearly defining the requirements springs from the observation that until today, to our knowledge, the attempts are characterized by a rigidity of views and no attempt has been conducted so far to unify them under a common vocabulary/standard. Hence, such a proposition should take into account the following aspects.

1. **Dynamic changing**. Any mapping should be dynamic as it will be subject to frequent changes. It should be easily updateable and, even better, it should be automatically updated. The vision is to provide a framework that is aware of potentially concurrent changes in both information layers and adjusts to them without human intervention. D2R MAP, D2RQ and Relational.OWL offer the creation of automated mappings; we should nevertheless comment on the absence of a runtime environment where changes in the data sources will be automatically reflected in the result mapping.

2. **User-centered collaborative design**. It is a well known fact that ontology authoring is not a linear procedure. Usually, an ontology is developed/authored by a group of people because it often spans over different conceptual areas. The requirement of collaborative design is emphasized when it concerns mappings of a database to an ontology, since every mapping statement demands from the author(s) knowledge of the domains of both the database and ontology models. Therefore, unless a collaborative approach is adopted, the viability of the system cannot be easily ensured. D2RQ, D2R MAP and R_2O offer descriptive languages (i.e. vocabularies) in which the mapping can be expressed and are thus allowing team work on mapping. We should also mention the API of the commercial tool RDF Gateway that allows the mapping procedure to be regarded as code authoring.

3. **Conformity with Standard formats**. They should conform to RDF, KIF, OWL and SQL. Moreover, it would be better to accept formats besides the RDF/XML notation. Turtle, N3/Notation3 and N-Triple serializations should also be supported. In [59] it is argued that not only the transmission format is important, but also the way the data is encoded. The tools surveyed are all based on standardized languages, however, we would favour more complete approaches. In Table 1, the columns ‘Ontology Language’, and ‘Semantic Query Language’ present the ampleness of the surveyed tools.

4. **Versioning and rollback** are crucial procedures, especially in team working. Since neither an ontology authoring nor a database designing procedure is a simple linear task, versioning should be offered in a way similar to CVS (Concurrent Versions System) or Microsoft VSS that is familiar to developers. The tools that offer a GUI constitute monolithic approaches since the user is limited to the capabilities of the graphical tool. Contrarily, non-graphical interfaces – descriptive languages such as D2RQ, D2R MAP, R2O or APIs such as RDF Gateway – allow concurrent processing, a matter that is crucial in commercial environments.

5. **Automation at design-time** is very important because manual annotation is typically an error-prone and time-consuming task. Companies in general are unwilling to spend resources in order to achieve doubtful results. These two arguments taken into account, the mapping procedure should be

automated to the maximum possible extent. The column 'Information needed/Degree of Automation' in Table 1 illustrates the automation level offered by the described tools.

6. **Completeness**. Any system should be able to provide freedom to the authors to capture any subset of the underlying information. In other words, anything should be able to be mapped to anything. Unfortunately, most of the current tools provide mappings between database tables and columns as shown in column 'Components mapped' in Table2. We would like to see tools that allow more general mapping definitions, such as arbitrary SQL queries.

7. **Reusability of the mappings.** The result of the mapping procedure should be easily extended and combined to other mappings. Developers should be able to reuse and consolidate fragments or the total of their previous work, enhancing its durability. A common mapping language standard – such as XMI for XML or KIF for ontologies – would provide an important evolution to this direction and would allow interoperability among Web engineers. However, the tools that are implemented as graphical applications (Clio, MOMIS, MAPONTO) comprise monolithic approaches that lack reusability. The remainder of the tools allows manual processing but, because of the absence of a common standard, the user remains restricted to using the specific tool.

We would like to point out as well that the Semantic Web scientific area should borrow ideas from the database community. First, there is a lot of experience in this field whose results should be integrated in the Semantic Web vision. Real world practice has brought up challenges and solutions that should be taken into account when designing the next steps. Second, there is a wide society in Web Engineering that is unwilling to modify legacy systems that are fully functional and covering current needs. Third, there is a lot of expertise in the domain of databases that would be preferable to integrate with, rather than migrate to new tools and technologies.

To sum up, in this paper, we attempted to gather together the most important and contributing approaches in the subject of database and ontology collaboration, or mapping as it is frequently cited in the text. We tried to provide the reader with a concise overview of these approaches, in a way that the comparison between them is easy and straightforward. This effort has revealed that there are mainly one or two main ideas, which most of the aforementioned approaches share and try to expand. Therefore, we dare to say that efforts in the field have become stagnant and that, maybe, a completely novel approach that takes into consideration only few of the points we have highlighted, will render the collaboration between ontologies and databases more efficient and simple for the end user. Hence, we believe that future progress in this research domain should be closely looked on and this is what we are planning to do, incorporating any innovative methods to this work and, of course, more tangible measures of comparison, such as extensive benchmarking.

References

1. M. K. Bergman (2001), *The Deep Web: Surfacing Hidden Value*, The Journal of Electronic Publishing Vol. 7, No. 1. Available online at http://www.press.umich.edu/jep/07-01/bergman.html, accessed on 16-11-2007.
2. T. Berners-Lee, J. Hendler and O. Lassila (2001), *Semantic Web - new form of Web content that is meaningful to computers will unleash revolution of new possibilities*, Scientific American, pp29-37.
3. T. R. Gruber (1995), *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, International Journal of Human-Computer Studies, Vol. 43, No. 5, pp. 907-928.

4. I. Kurtev, J. Bézivin and M. Aksit (2002), *Technological spaces: An initial appraisal*, in 10th International Conference on Cooperative Information Systems (CoopIS 2002), International Symposium on Distributed Objects and Applications (DOA 2002), Federated Conferences.
5. G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, W. Schwinger and M. Wimmer (2006), *Lifting metamodels to ontologies: A step to the semantic integration of modeling languages*, in Proceedings of Model Driven Engineering Languages and Systems (MoDELS 2006), Vol. 4199 of Lecture Notes in Computer Science, Springer, pp. 528-542.
6. F. S. Parreiras, S. Staab and A. Winter (2007), *On marrying ontological and metamodeling technical spaces*, in Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering.
7. M. R. Genesereth and N. J. Nilsson (1987), *Logical Foundation of Artificial Intelligence*, Morgan Kaufmann Publishers.
8. A. Borgida, M. Lenzerini and R. Rosati (2002), *Description Logics for Databases*, in the Description Logic Handbook, edited by F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, Cambridge University Press, pp. 472-494.
9. D. Beckett (2004), *RDF/XML Syntax Specification (Revised)*, http://www.w3.org/TR/rdf-syntax-grammar/, accessed on 20-11-2007.
10. O. Lassila and R. R. Swik (1999), *RDF model and syntax specification*, W3C Recommendation, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222, accessed on 18-11-07.
11. I. Horrocks, P. Patel-Schneider and F. van Harmelen (2003), *From $\mathcal{SHIQ}$ and RDF to OWL: the making of a Web Ontology Language*, Journal of Web Semantics, Vol. 1, No. 1, pp. 7-26.
12. D. McGuinness and F. van Harmelen (2004), *OWL Web Ontology Language Overview*, http://www.w3.org/TR/owl-features/, accessed on 18-11-2007.
13. P. F. Patel-Schneider and I. Horrocks (2004), *OWL Web Ontology Language: Semantics and Abstract Syntax Section 3. Direct Model-Theoretic Semantics*. Available online at http://www.w3.org/TR/owl-semantics/direct.html#3.1, accessed on 12-11-2007.
14. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur and Y. Katz (2007), *Pellet: A practical OWL-DL reasoner*, Journal of Web Semantics, Vol. 5, No. 2, pp. 51-53.
15. D. Tsarkov and I. Horrocks (2005), *Ordering Heuristics for Description Logic Reasoning*, in Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), pp. 609–614, Morgan Kaufmann Publishers.
16. B. Motik and U. Sattler (2006), *A comparison of reasoning techniques for querying large description logic Aboxes*, in Proceedings of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006).
17. V. Haarslev and R. Möller (2001), *RACER System Description*, in Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR 2001), Vol. 2083 of Lecture Notes in Artificial Intelligence, pp. 701–706, Springer.
18. S. Bechhofer (2006), *DIG 2.0: The DIG Description Logic Interface, DIG Working Group Note*, http://dig.cs.manchester.ac.uk/, accessed on 21-11-2007.
19. F. Baader and W. Nutt (2002), *Basic Description Logics*, in the Description Logic Handbook, edited by F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, P. F. Patel-Schneider, Cambridge University Press, pp. 47-100.
20. A. Ramakrishnan, R. Gehrke (2003), *Database Management Systems* 3rd Edition, McGraw Hill, Chapter 3: The Relational Model, pp. 57-99.
21. E. F. Codd (1970), *A relational model of data for large shared data banks*, Communications of the ACM, Vol. 13, No. 6, pp. 377–387.
22. Object Management Group (2002), *MOF 2.0 Query/Views/Transformations RFP*, http://www.omg.org/cgi-bin/doc?ad/2002-4-10, accessed on 6-11-2007.
23. E. Prud'hommeaux and A. Seaborne (2005), *SPARQL Query Language for RDF*, http://www.w3.org/TR/rdf-sparql-query/, accessed on 18-11-2007.

24. J. Bailey, F. Bry, T. Furche and S. Schaffert (2005), *Web and Semantic Web Query Languages: A Survey*, in Reasoning Web, First International Summer School 2005, edited by N. Eisinger and J. Małuszynski, Vol. 3564 of Lecture Notes in Computer Science, Springer-Verlag.
25. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis and M. Scholl (2002), *RQL: A declarative query language for RDF*, in Proceedings of the 11th International World Wide Web Conference (WWW 2002).
26. J. Broekstra, A. Kampman and F. van Harmelen (2003), *Sesame: An architecture for storing and querying RDF data and schema information*, in Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, edited by D. Fensel, J. A. Hendler, H. Lieberman and W. Wahlster, MIT Press, pp. 197–222.
27. C. P. de Laborda and S. Conrad (2006), *Bringing relational data into the Semantic Web using SPARQL and Relational.OWL*, in Proceedings of the International Conference on Data Engineering Workshops (ICDEW 2006), pp. 55.
28. S. Harris and N. Gibbins (2003), *3store: Efficient Bulk RDF Storage*, in Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS 2003), pp. 1-15.
29. O. Corby, R. Dieng-Kuntz and C. Faron-Zucker (2004), *Querying the Semantic Web with the Corese search engine*, in Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2004), Prestigious Applications of Intelligent Systems (PAIS 2004), pp. 705-709.
30. D. Wood, P. Gearon and T. Adams (2005), *Kowari: A Platform for Semantic Web Storage and Analysis*, in Proceedings of the XTech Conference.
31. I. Horrocks, L. Li, D. Turi, and S. Bechhofer (2004), *The Instance Store: Description Logic Reasoning with Large Numbers of Individuals*, in 17th International Workshop on Description Logics (DL 2004), pp. 31-40.
32. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini (2005), *Building a Tourism Information Provider with the MOMIS System*, in Journal of Information Technology & Tourism, Vol. 7, No. 3-4, pp. 221-238.
33. S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano (2001), *Semantic Integration of Heterogeneous Information Sources*, in Journal of Data & Knowledge Engineering, Vol. 36, Elsevier Science B.V., pp. 215-249.
34. L. Haas, M. Hernandez, H. Ho, L. Popa, and M. Roth (2005), *Clio Grows Up: From Research Prototype to Industrial Tool*, in ACM SIGMOD International Conference on Management of Data, pp. 805-810.
35. L. Yan, R. Miller, L. Haas, and R. Fagin (2001), *Data-Driven Understanding and Refinement of Schema Mappings*, in International Conference on the Management of Data, ACM SIGMOD Vol. 30, pp. 485-496.
36. D. Maier, A. Mendelzon, and Y. Sagiv (1979), *Testing Implications of Data Dependencies*, in ACM Transactions on Database Systems (TODS), Vol. 4, No. 4, pp. 455–469.
37. L. Stojanovic, N. Stojanovic, and R. Volz (2002), *Migrating Data-intensive Web sites into the Semantic Web*, in ACM Symposium on Applied Computing (SAC 2002), pp. 1100-1107.
38. C. Bizer (2003), *D2R MAP – A Database to RDF Mapping Language*, in 12th International World Wide Web Conference (WWW 2003).
39. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson (2003), *Jena: Implementing the Semantic Web recommendations*, Technical Report (HPL- 2003-146), Hewlett-Packard.
40. Hewlett-Packard Development Company, LP (2006), *Jena - A Semantic Web Framework for Java*, http://jena.sourceforge.net/, accessed on 18-11-2007.
41. C. Bizer and A. Seaborne (2004), *D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs*, in the 3rd International Semantic Web Conference (ISWC 2004).
42. J. Barrasa, O. Corcho, and A. Gómez-Pérez (2003), *Fund Finder Wrapper: A Case Study of Database-to-ontology Mapping*, in International Workshop on Semantic Integration, pp9-15.

43. J. Barrasa, O. Corcho, and A. Gómez-Pérez (2004), *R_2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language*, in 2nd Workshop on Semantic Web and Databases.
44. E. Bozsak et al. (2002), *KAON - Towards a large scale Semantic Web. E-Commerce and Web Technologies*, in 3rd International Conference on Electronic Commerce and Web Technologies (EC-Web 2002), Vol. 2455 of Lecture Notes in Computer Science, Springer, pp. 304-313.
45. R. Volz, S. Handschuh, S. Staab, L. Stojanovic, and N. Stojanovic (2004), *Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the Semantic Web*, in Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 1, No. 2, pp187–206.
46. D. Dou, and P. LePendu (2006), *Ontology-based Integration for Relational Databases*, in the 21st ACM Symposium on Applied computing (SAC 06), pp. 461-466.
47. D. Dou, J. Pan, H. Qin, and P. LePendu (2006), *Towards Populating and Querying the Semantic Web*, in 2nd Int Workshop on Scalable Semantic Web Knowledge Base Systems, pp129-142.
48. R. Fikes, P. Hayes, and I. Horrocks (2005), *OWL-QL – A language for deductive query answering on the Semantic Web*, in Journal of Web Semantics, Vol 2, No. 1, pp. 19-29.
49. Y. An, A. Borgida, and J. Mylopoulos (2006), *Building Semantic Mappings from Databases to Ontologies*, in the 21st National Conference on Artificial Intelligence (AAAI 06).
50. N. Konstantinou, D. Spanos, M. Chalas, E. Solidakis, and N. Mitrou (2006), *VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents*, in International Workshop on Web Information Systems Modeling (WISM 06), pp. 36-47.
51. Y. Kalfoglou, B. Hu, and D. Reynolds (2005), *On Interoperability of Ontologies for Web-based Educational Systems*, in Intl Workshop on Interoperability of Web-based Educational Systems.
52. R. Cyganiak, *Benchmarking D2RQ v0.2*, available online at http://sites.wiwiss.fu-berlin.de/suhl/bizer/ D2RQ/benchmarks/index.html, accessed on 18-11-2007.
53. P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krötzsch, L. Serafini, G. Stamou, Y. Sure, and S. Tessaris (2004), *Specification of a common framework for characterizing alignment*, KnowledgeWeb deliverable d2.2.1v2. Technical report, Institut AIFB, Universität Karlsruhe.
54. N. Choi, I. Y. Song and H. Han (2006), *A Survey on Ontology Mapping*, SIGMOD Record, Vol. 35, No. 3, pp. 34-41.
55. Y. Kalfoglou and M. Schorlemmer (2003), *Ontology mapping: the state of the art*. The Knowledge Engineering Review, Vol. 18, No. 1, pp. 1-31.
56. I. Niles and A. Pease (2001), *Towards a Standard Upper Ontology*, in Proc. of 2nd International Conference on Formal Ontology in Information Systems, edited by C. Welty and B. Smith.
57. L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi and J. Sachs (2004), *Swoogle: A search and metadata engine for the Semantic Web*, in Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM 2004), pp. 652-659.
58. I. Niles and A. Pease (2003), *Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology*, in Proceedings of the International Conference on Information and Knowledge Engineering (IKE 2003), Vol. 1, CSREA Press, pp. 412-416.
59. T. Weithoner, T. Liebig, M. Luther, S. Bohm, F. von Henke and O. Noppens (2007), *Real World Reasoning with OWL*, in Proc. of 4th European Semantic Web Conference, Vol. 4519, Springer, pp. 296-310.
60. Y. Guo, Z. Pan and J. Heflin (2004), *An Evaluation of Knowledge Base Systems for Large OWL Datasets*, in Proc. of the 3rd International Semantic Web Conference (ISWC 2004), pp. 274–288.
61. L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan and S. Liu (2006), *Towards a Complete OWL Ontology Benchmark*, in Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Vol. 4011 of Lecture Notes in Computer Science, Springer, pp. 125–139.
62. V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta and F. Ciravegna (2006), *Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art*, Journal of Web Semantics Vol. 4, No. 1, pp. 14-28.