VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents

Nikolaos Konstantinou, Dimitrios-Emmanuel Spanos, Michael Chalas, Emmanuel Solidakis and Nikolas Mitrou

nkons@cn.ntua.gr, el00057@mail.ntua.gr, el00114@mail.ntua.gr, esolid@telecom.ntua.gr, mitrou@softlab.ntua.gr

National Technical University of Athens, Division of Communications, Electronics and Information Systems, Heroon Polytechniou 9, 15773 Athens, Greece

Abstract. This paper introduces an approach to mapping relational database contents to ontologies. The current effort is motivated by the need of including into the Semantic Web volumes of web data not satisfied by current search engines. A graphical tool is developed in order to ease the mapping procedure and export enhanced ontologies linked to database entities. Moreover, queries using semantic web query languages can be imposed to the database through its connection to an ontology. Using Protégé, we were able to map ontology instances into relational databases and retrieve results by semantic web query languages. The key idea is that, instead of storing instances along with the ontology terminology, we can keep them stored in a database and maintain a link to the dataset. Thus, on one hand we achieve smaller size ontologies and on the other hand we can exploit database contents harmonizing the semantic web concept with current wide-spread techniques and popular applications.

1 Introduction

It is a well known fact that the rapid evolution of the Internet has brought up significant changes to information management. The web of knowledge changed the way people share, access and retrieve data. The existing data that lie on the web offer a significant source of information for almost anything.

Search engines have been evolved and research has been conducted in order to exploit the web resources. Since Internet became a public common currency and mostly referred to as the World Wide Web, its content growth made the use of search engines a gateway to information. Without Google, Yahoo! or Altavista the web as it is today would be useless. Terabytes of data in millions of web pages are impossible to be searched without the use of special Information Retrieval (IR) techniques offered by the current search engine implementations.

Nevertheless, still much work needs to be done so as to render all this information retrievable. It is well known that there is a large quantity of existing data on the web stored using relational database technology. This information is often referred to as the Deep Web [7] as opposed to the Surface Web comprising all static web pages.

Deep Web pages don't exist until they are generated dynamically in response to a direct request. As a consequence, traditional search engines cannot retrieve their content and the only manageable way of adding semantics to them is attacking directly its source: the database.

The creator of the web, Tim Berners-Lee proposed the idea of the 'Semantic web' to overcome the handicaps referenced. As stated in [1], 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. This process will consume much effort. Researchers have developed Semantic Web tools to facilitate this effort.

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 point of the idea. Ontologies provide conceptual domain models, which are understandable to both human beings and machines as a shared conceptualization of a specific domain that is given [6]. With the use of ontologies, content is made suitable for machine consumption, contrary to 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. The application developed and presented in this paper introduces a solution to this problem. Current data is and should be maintained separately while ontologies can be developed in order to enhance its meaning and usefulness.

The paper is organized as follows: Section 2 provides the definitions of the aspects discussed in this paper. Section 3 details the mapping architecture, the mapping process and the integrity checks performed to assure consistent results. In Section 4 we present our case study and compare it to related efforts in Section 5. We conclude and discuss about future work in Section 6.

2 Definitions

This section provides a background definition of our work. This will make clearer every aspect of the proposed idea.

2.1 Description Logics Ontologies

Formally, an ontology can be defined as a model of a knowledge base (KB). 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. A KB created using Description Logics comprises the two following components [16].

The first part contains all the concept descriptions, the intentional knowledge and is called Terminological Box (TBox). The TBox introduces the terminology, 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 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.

2.2 Relational Models

According to [4], a database is a collection of relations with distinct relation names. The relation 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 though 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).

In order to provide access to their contents, databases support many query languages, but SQL is the practical choice. SQL 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.

2.3 Mapping Relational Database Contents to Ontologies

In a simplified point of view, we can compare the schema of a relational database to the TBox of a KB and the instance to the actual data to the ABox. We could claim that databases are similar to Knowledge Bases because of the fact that they both are used to maintain models and data of some domain of discourse (UofD) [17]. 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 our work, we succeeded in mapping the relational database contents to the TBox of the ontology. The description language chosen is OWL-DL [2, 5], based on standard predicate calculus. The ontology produced does not contain an ABox, only a reference to the dataset in the database. The dataset can be any data combination, not just table tuples. A visualization of the mentioned work is shown in the figure below.

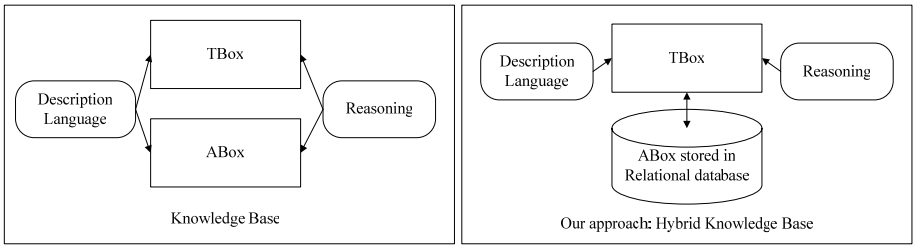

Fig. 1. Main Idea. The Assertion Box of the Knowledge Base is kept separately in a relational database. The resulting hybrid ontology will contain references to database datasets that comprise the class individuals

What has been accomplished in the current work is to reference database tuples through the ontology TBox. Common approaches describe mapping mechanisms between ontology classes and database tables. Our enhancement from the scope of the Semantic Web lies on the fact that the mappings are capable of corresponding class individuals (alt. instances) to any possible dataset combination. From the view of database theory, we enriched the database schema structure to include description logics and answer queries on a semantic level.

2.4 Issues and Limitations Rising from the Mapping Process

There are significant differences between the knowledge bases and databases. Among the differences is the 'open-world' semantics in opposition to the 'closed-world' semantics that hold respectively for each one.

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 told 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.

On the other hand, a query on a KB can return three types of answers that are true, false and 'cannot tell'. The 'open world assumption' states that lack of knowledge does not imply falsity. So a question in an appropriate schema 'Is Athens a capital city?' will return 'cannot tell' if the schema is not informed while a database schema would clearly state that 'no'.

Class properties were not mapped to relations between database tables. The reason is that regarding the ontology semantics, the class hierarchy is fully different from the property hierarchy. Foreign key constraints involving relations, though, can be mapped to Object Properties linking OWL classes. This approach however, would not produce more powerful results; it would rather complicate reasoning checks. So, in the work that is presented here, concept integrity checks are being held only during the mapping process without being an extra burden to the resulting ontology.

3 Mapping Architecture and Process

In this chapter we clarify the architecture of the technique through which the target goal is reached. We describe the mapping process, analyze the integrity constraints and explain how the user can retrieve database data through the intermediate mapping ontology.

3.1 Mapping Process

Our proposed mapping method consists of four steps. First, we capture data from the database. The data can be any combination of datasets. More particularly, the ability is given to select the desired subset of records belonging to various tables. In step two, we select a class of the ontology. Step three is the most important as all consistency checks are being performed in this step along with evaluation, validation and refinement of the mapping. In step four takes place the modification of the resulting ontology and the additional information is added to it. These four steps can be repeated as many times as required. By saving the mapping in a new ontology, additional information will be kept such as the database type and name, the initial ontology name and when the mapping is last modified.

The final result is the initial ontology enhanced to include references to datasets. These references will be under the form of class properties in the ontology, all assigned as value a string containing the actual SQL query that returns the dataset. Each query posed to the ontology will check the existence of the mapping property. In the case that it does not exist, the results are no affected. Otherwise, the results will be database entries instead of class individuals.

3.2 Validation of the Mapping Process – Consistency checks

During the mapping process, tests are run to insure the concept's consistency. These tests require enough computational time and are therefore computed once, responding with a positive or negative answer, informing the user in each case. Every time a new mapping is defined by the user, the following requirements must be met.

First of all, two disjoint classes cannot have mappings to the database that contain common records or subsets of records. Even in the case of finding a single datum in common, the mapping will be rejected. We note that 'common data' will as well be considered data that belong to different fields of two tables connected with a foreign key relationship. Disjoint classes cannot be mapped to records of these fields because a foreign key connects the two tables semantically. The classes' disjointment does not allow such connections. This constraint reassures that the datasets will be as well disjoint. In addition, the same check is run respectively for the class' sub- and superclasses.

Moreover, a check is performed to reassure that subclass hierarchy is maintained in the mapped data. More specifically, if a class A is subclass of a class B and the two classes are mapped to record sets R(A) and R(B) respectively, then R(A) must be a

subset of R(B) as well. This check must hold recursively for every mapping statement. Special care is taken of table fields that have foreign key relationships. Subclasses can be mapped to field entries outside the domain of the superclass mapping as long as these fields are referred to by foreign keys. In conclusion, for a mapping to be accepted, the set of data for each class must be a subset of the class' superclasses.

The checks are performed after each mapping command. The checks were intentionally not strict as the system was designed keeping in mind the final user: the domain expert who will produce the mapping result. Thus, only the necessary consistency checks were implemented balancing the user's freedom with the concept's integrity.

3.3 Semantic Query Execution

Our implementation includes the functionality of imposing queries by terms of semantic web query languages. Fig. 3 illustrates how the application handles user's queries.

First, users' queries are parsed and checked syntactically. The query would instantly return with the desired results but here is where the mapping intervenes. Instead of returning the class resources, for each class we check if there exists the mapping property. If the related property is found, the query is redirected to the database. The results are then formatted and served to the user.

The achievement lies in the fact that we added an extra layer of interoperability, between the application GUI and the actual data. This hybrid approach leaves the ontology and the actual data intact, only serving as a semantics addition to current systems. The generalization of the current concept to current web technologies is about to offer a semantic layer that will add the desired intelligence without modifying legacy systems.

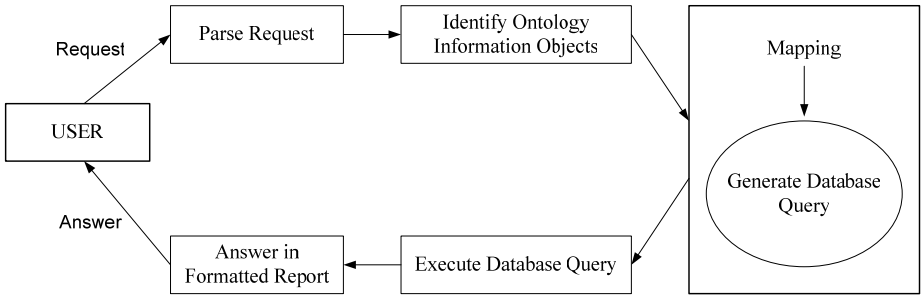

Fig. 3. After the generation of the mapping ontology, user's queries are processed invisibly by the database mapping mechanism

3.4 Implementation-Specific Information

Our implementation was built as a plug-in on top of the Protégé [3] ontology editor. Protégé was chosen among other open-source ontology editors like SWOOP [18],

SMORE [19], OilEd [20] or Ontolingua [21] because of its extensible structure and documentation offered about writing a plug-in. Protégé is java-based and supports ontologies authored in OWL.

For a database backend, the VisAVis tool supports connections via JDBC to the MySQL and PostgreSQL database management systems.

As far as it concerns the Semantic Web query languages, several recommendations have been considered. The issue is that query languages about OWL ontologies are still in their infancy [22]. We would therefore have to select between the SPARQL and RQL families of query languages for RDF. The practical choice is RDQL [15] as it is implemented and fully supported in the Jena [12] framework upon which Protégé is built. RDQL has recently been submitted to the W3C for standardization (October 2003).

4 The 'VisAVis' Use Case Scenario

In this section, we give a practical example of the mapping method described above. For our example's purpose, we will use an OWL-DL ontology that can be found at [3] and includes some terms concerning tourism, such as lodgings, activities and destinations. Our aim is to map the classes of the ontology to data that refer to the same concept as the ontology classes. That is why, in this example, we are using as well a database containing contextual data.

We have implemented the mapping method described in the previous section, as a Protégé tab plug-in, colorfully named as VisAVisTab and shown in Fig. 4. The first step is to open the ontology that contains the classes of interest. Then, we connect with the corresponding database that contains the real data. At that point, we can see both the ontology and the database hierarchy and we are able to select a set of data from the database using the application's graphical user interface. In fact, the application constructs, in the background, an SQL query that, when executed, returns the desired dataset. This SQL query, and not the data set itself, will be correlated with a certain ontology class selected by the user.

To demonstrate VisAVis in action, we are using a database called 'travel' that contains among others the tables 'cities', 'hotels', 'museums' and 'activities'. The table 'activities' contains information about activities offered to tourists, including its description and its type (surfing, hiking etc.). The activities' types are stored in the 'activity_type_id' column, which is foreign key to the 'id' column of the 'activities_types' table.

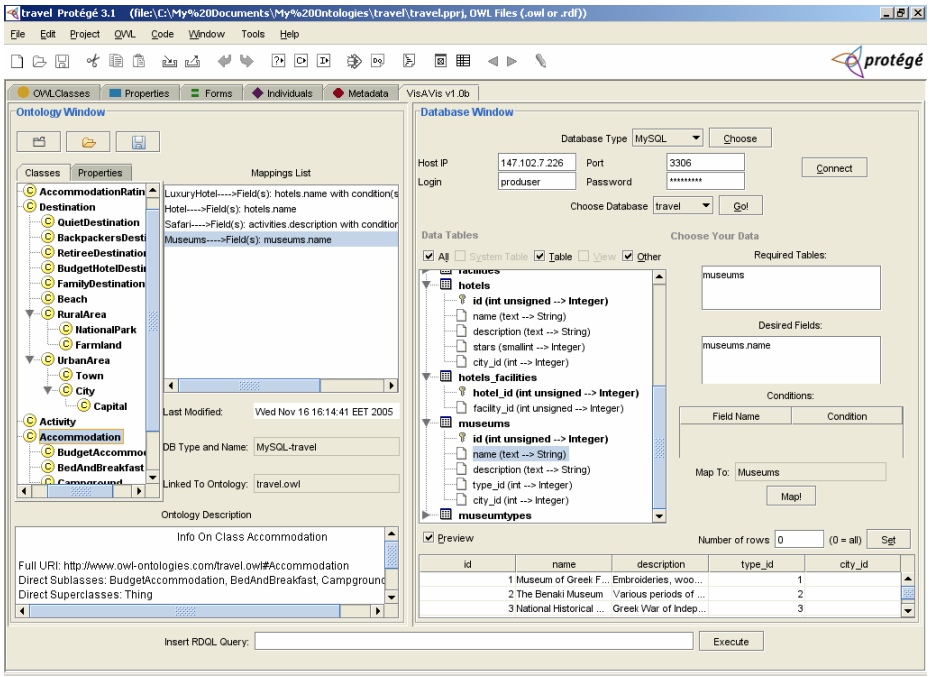

Fig. 4. The VisAVis Protégé Tab. The left panel contains the class hierarchy, the mapped classes list and information about the selected class as well as for the ontology itself. The right panel is dedicated to the database. The user can select multiple tables and fields, declare possible mapping conditions and confirm mappings. At the bottom of the screen, RDQL queries can be executed

A common approach would be to map the class 'Activity' of the OWL ontology to the columns 'id', 'description', 'activity_type_id' from the 'activities' table and to the 'id', 'name', 'description' columns of the 'activities_types' table. The data selection process is easily carried out by dragging the columns needed from the database hierarchy and dropping them into the 'Desired Fields' field. In order to map the class 'Hiking', which is a subclass of the 'Activity' class, we should add a condition that would enable us to select only the activities of the certain type. Likewise, the classes 'Surfing' and 'Safari', which are also subclasses of the 'Activity' class, could be mapped to the appropriate data that would satisfy the necessary conditions. The conditions can be applied to a certain set of columns by dragging the necessary column, dropping it to the 'Conditions' field and then, filling in the rest of the constraint. When the data selection has finished, the user can complete the mapping process by pressing the 'Map!' button.

It is worth mentioning that when columns from two or more tables are selected, our application includes in the SQL query an additional condition representing the referential integrity constraint. In other words, foreign keys are taken into account whether they are user-specified or not and added to the final SQL query. This happens even in the case where two or more tables have two or more distinct relations

between them. For instance, for the following example a mapping concerning the class ‘Activity’ is the following:

```
  Hiking -> Field(s): activities.description with condi-
tion(s): (activities_types.name=’Hiking’)
```

The corresponding SQL query, which is automatically generated by our SQL builder, is shown below in a snapshot of the enhanced ontology:

```
<owl:Class rdf:about="#Hiking">
    <queryString>SELECT activities.description FROM
    activities, activities_types WHERE
    (activities.activity_type_id=activities_types.id)
    AND (activities_types.name = "Hiking")
    </queryString>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Sports"/>
    </rdfs:subClassOf>
</owl:Class>
```

As we notice in the above sample, the mapping process has as a result the introduction of a datatype property to the class. The property is arbitrarily named ‘queryString’ and is given as domain the class’ domain and as value the corresponding SQL query. Thus, the initial OWL ontology is enhanced with additional properties that represent the mappings accomplished. Following the same course of actions, we can map the entire ontology to sets of data from the database.

We should also note that during the mapping process, consistency checks are executed in order to check whether mappings conform to the ontology rules or not as described in Section 3.2. For instance, if we tried to map to the class ‘Activity’ some data from the ‘hotels’ table, the mapping validator would reject this mapping as inconsistent with the ontology rules, since classes ‘Activity’ and ‘Accomodation’ are disjoint.

Finally, in order to test the whole mapping process, our application enables the execution of a semantic query, which will return actual data from the database, instead of ontology elements. As it has already been mentioned, a similar approach could be followed in a sophisticated Semantic Web search engine, that would first execute a semantic query and then, using mappings between concepts and sets of actual data or documents, it would return these sets of data or documents.

5 Related Work

The Semantic Web is the new WWW infrastructure that will enable machine process of the web content and seems to be a solution for many drawbacks of the current Web. The semantic information addition to the current web will create a new information layer on top of the current technologies. Ontology to database mapping is an active research topic to this direction based on the observation that the web content is

dynamic and most of it originates from databases. Several approaches have been presented aiming at semantic integration proposing techniques for database-to-ontology correspondence. Related work can be divided in two fields, database-to-ontology migration and database-to-ontology collaboration. Our approach belongs to the latter; hence similar work will be given reference in the current chapter.

The Ontomat Application Framework introduced in [9] 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 [10] is part of the framework and offers the means to semi-automatically realize mappings 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.

D2RMap was first presented in [11] and provides means to declaratively state ontology-to-database mappings. D2R is based on XML syntax and constitutes a language provided to assign ontology concepts to database sets. The result includes SQL commands directly to the mapping rules and is capable of performing flexible mappings. The mapping procedure in D2R is similar to ours. Results are extracted in RDF, N3, RDF or Jena [12] models.

R_2O [14] is another declarative language and serves the same purpose as it is obvious in the full title, Relational 2 Ontology. The R_2O language is fully declarative and extensible while expressive enough to describe the semantics of the mappings. In general, like D2R, it allows the definition of explicit correspondences between components of two models.

The Unicorn workbench [13] was developed aiming at the semantic integration which is as well the concept of our work. Unicorn's architecture is two-layered meaning that the external assets layer is separated from the model-information layer.

6 Conclusions and Future Work

We have developed a novel, integrated and semi-automated approach for mapping and reusing large sets of data into the Semantic Web. Using the tool described above, it is possible to create a mapping between any relational database and any ontology. For the mapping to have a meaning, however, it is essential that the concepts described are at some point similar. For example, it has no meaning mapping a database containing a company's customers to an ontology describing plants. The logical aspect is left to the judgment of the user.

The current version implements a rather small subset of the SQL language. The mapping to the actual data is in fact accomplished using 'select from where' functions. We are currently working on enriching the ontology result to include more functions such as 'group by' and 'order by'. Functions such as max(), count() and avg() are not supported either, in the current version. Support needs to be added for several RDBMS such as Oracle or Microsoft SQL Server to cover a significant percentage of today's choices. Yet, these topics are only implementation-specific and of no special research interest. They are however needed if the aim is to provide the Protégé plug-in the tool's corresponding plug-in section.

An important open issue in the current implementation is also the fact that queries only return results from the database. The user should also be informed about class individuals, not only database records. The current development is not purposed to distinct between ontologies and databases; it is about offering a collaboration framework and an intermediate layer for database exploitation by Semantic Web applications.

The resulting enhanced ontology is subject to easy programmatic access. In addition, once the mapping result is created, our tool will only be needed in case of change of the database schema. Practice has proven that database schemas are not frequently changing. Our intention is to use such ontologies combined with software agents in order to create an intelligent environment where data search and retrieval will implement the Semantic Web concept: users will be able to search among large volumes of semantically annotated data.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, 2001.
2. I. Horrocks, P. Patel-Schneider, F. van Harmelen: From SHIQ and RDF to OWL: the making of a Web Ontology Language.
3. The Protégé Ontology Editor and Knowledge Base Framework, http://protege.stanford.edu/.
4. Ramakrishnan and Gehrke: Database Management Systems 3rd Edition, McGraw Hill 2003, Chapter 3: The Relational Model, pages 57-99.
5. OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/.
6. Gruber, 1993: Toward principles for the design of ontologies used for knowledge sharing.
7. Bergman MK. *The Deep Web: Surfacing hidden value*. White paper. Sept 2001.
8. Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387
9. Erol Bozsak et al: KAON - Towards a large scale Semantic Web. *E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*, volume 2455 of Lecture Notes in Computer Science, pp. 304-313. Springer, 2002.
10. *R. Volz et al: 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 (2004) 187–206.*
11. Christian Bizer: D2R MAP – A Database to RDF Mapping Language, *The twelfth international World Wide Web Conference, WWW2003, Budapest, Hungary*.
12. Jena: A Semantic Web Framework for Java, http://jena.sourceforge.net/.
13. Jos de Bruijn: Semantic Integration of Disparate Data Sources in the COG Project, in *Proceedings of the International Conference on Enterprise Information Systems (ICEIS2004)*, Porto, Portugal, 2004.
14. Barrasa J, Corcho O, Gómez-Pérez: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. A. *Second Workshop on Semantic Web and Databases (SWDB2004)*, Toronto, Canada. August 2004.
15. A. Seaborne, RDQL – RDF Data Query Language, part of the Jena RDF Toolkit, HPLabs Semantic Web activity, http://hpl.hp.com/semweb/, RDQL grammar: http://www.hpl.hp.com/semweb/rdql-grammar.html, Online only.

16. Franz Baader, Werner Nutt: 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, 2002, pages 47-100.
17. A. Borgida, M. Lenzerini, R. Rosati. 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, 2002, pages 472-494.
18. Aditya Kalyanpur, Bijan Parsia, James Hendler: A Tool for Working with Web Ontologies, In *Proceedings of the International Journal on Semantic Web and Information Systems, Vol.1, No.1, Jan-Mar 2005*.
19. Aditya Kalyanpur, Bijan Parsia, James Hendler, and Jennifer Golbeck. SMORE - semantic markup, ontology, and RDF editor.
20. Sean Bechhofer, Ian Horrocks, Carole Goble, Robert Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *DL2001, 14th International Workshop on Description Logics, Stanford, USA, August 2001*.
21. A. Farquhar, R. Fikes, & J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Knowledge Systems, AI Laboratory, 1996.
22. James Bailey, François Bry, Tim Furche, and Sebastian Schaffert: Web and Semantic Web Query Languages: A Survey, In: *Reasoning Web, First International Summer School 2005*, Norbert Eisinger and Jan Małuszynski, editors. Springer-Verlag, LNCS 3564, 2005.