Uncovering the Deep Web: Transferring Relational Database Content and Metadata to OWL Ontologies

Damir Jurić, Marko Banek, and Zoran Skočir

Faculty of Electrical Engineering and Computing, University of Zagreb,
Unska 3, HR-10000 Zagreb, Croatia
{damir.juric,marko.banek,zoran.skocir}@fer.hr

Abstract. Organizing the publicly available Web content into highly systematized domain ontologies is a necessary step in the evolvement of the Semantic Web. A large portion of that content called the deep Web is stored in relational databases and it is not accessible to Web search engines. Incorporation of the deep Web data results in domain ontologies richer both in content and in semantic relations. In this paper we introduce a framework for an automatic mapping of relational database metadata and content to domain ontologies written in OWL. Relational constructs: relations, attributes and primary-foreign key associations are translated to OWL classes, datatype properties and object properties. Database tuples become ontology instances. In order to define reference points for integration with other ontologies the constructed ontologies are further enriched with additional semantics from the WordNet lexical database using word sense disambiguation mechanisms. A software implementation of the approach has been developed and evaluated on case study examples.

Keywords: ontology, OWL, relational database, deep Web, WordNet, word sense disambiguation.

1 Introduction

The scientific community has universally recognized the Semantic Web [1] as the prospective evolvement direction of the current Web. The core of the Semantic Web are ontologies written in OWL [2], which formalize the domains by defining classes and their properties and by assigning individuals to the classes.

Web search engines can access only the HTML pages, the "surface" of the Web, while much larger quantity of data, the deep Web [3], remains hidden: the data is available to the users but the pages do not exist until they are created dynamically as the result of a specific search.

The extraction of deep Web data (i.e. the content and the metadata of databases used to generate the pages) is a twofold contribution to creating the circumstances that will lead to the evolvement of the Semantic Web. First, domain ontologies that incorporate the *database content* are much richer than those including only the HTML page content. Second, relational *database metadata* provide additional semantic knowledge that can successfully be transferred to ontologies. This knowledge is lost once the Web pages are created and can thus be obtained only by accessing the databases directly. Uncovering

I. Lovrek, R.J. Howlett, and L.C. Jain (Eds.): KES 2008, Part I, LNAI 5177, pp. 456–463, 2008.

the deep Web causes no security or privacy risks to organizations willing to participate in Semantic Web projects, either by disclosing relational schema metadata or by exposing the already publicly available database content as a whole (i.e. as an ontology), instead in small portions (dynamically created HTML pages).

In this paper we introduce a framework for automatically creating OWL ontologies from extracted relational database metadata and content. We first describe a set of transformations that translate the relational database schema into ontology classes, properties and constraints and then add the database content as ontology instances. One contribution of our work is to apply word sense disambiguation mechanisms to acquire additional semantics from the large spectrum of semantic relations in the WordNet lexical database [4]. Java-based prototype software that evaluates the presented approach on case study examples is another major contribution of the paper.

The paper is structured as follows. Section 2 gives an overview of the related work. Rules for mapping relational metadata to OWL structures are explained in Section 3. Section 4 presents the process of adding additional semantics to the constructed ontologies and gives its evaluation. The architecture of the prototype software tool is illustrated in Section 5. Conclusions are drawn in Section 6.

2 Related Work

While many existing works focus on creating entity-relationship or object-based models from relational databases, only few solutions deal with automatic extraction of database semantics. However, none of them uses a standard ontology language. In [5] ontologies are generated semi-automatically from relational databases using Frame-Logics. The main drawback is the fact that a continuous interaction with the user is required. Similarly, in [6] the primary focus is kept on analyzing key, data and attribute correlations, as well as their combinations. Basic concepts of automatic reverse engineering are considered in [7]. Relations are mapped as classes, their attributes as class attributes and tuples as ontology instances. The primary-foreign key mechanism and specialization are not considered. In [8] we sketched the principles for transforming relational database metadata into OWL structures, but without discussing attribute constrains. Besides, neither a solution for acquiring additional semantics for ontology integration was proposed, nor a software implementation developed. A different approach is presented in [9], where OWL ontologies are created semi-automatically, corresponding to the content of the relational database and based on analyzing the resulting HTML forms.

A detailed overview of the most up-to-date approaches to ontology integration is given in [10]. Providing an unambiguous meaning of ontology components (particularly classes) is regarded as a necessary step to eliminate false matches caused by homonymy of terms. Annotation of the ontology components to be matched with entries from a background ontology with a comprehensive coverage of the domain of interest of the match target ontologies (such as WordNet) is considered necessary for the disambiguation of multiple possible meanings of terms. However, in most of the works annotations are either created manually or supposed to be provided earlier [11, 12]. The ontology matching approach described in [13] provides disambiguation of terms to a certain extent. Disambiguation i.e. choosing the right sense for a word in its

occurring context is based on statistical analysis of ontology instances. On the contrary, we apply disambiguation techniques based exclusively on dictionary data.

3 Rules for Mapping Relational Database Schema Components

The process of generating an ontology from a relational database consists of two phases. In the first phase a relational database is translated to an OWL ontology, while in the second phase WordNet is used to enrich the ontology with additional semantics. The first phase is explained in the remainder of this section. The COMPANY database [8] is used as an example to illustrate the mapping procedure (Fig. 1). Primary key attributes are underlined.

Mapping a relational database to an ontology is based on analyzing both the schema metadata (keys and attributes) and the content. The process of mapping a relational schema to OWL consists of several actions performed in the following order: (1) mapping relations, (2) mapping attributes, (3) mapping primary keys, (4) mapping 1:1 and N:1 binary relationships.

```
EMPLOYEE (Fname, Minit, Lname, Ssn, Bdate, Address, Sex,
          Salary, Super_snn, Dnumber)
DEPARTMENT (Dname, Dnumber, Mgr_ssn, Mgr_start_date)
DEPT_LOCATIONS (Dnumber, Dlocation)
PROJECT (Pname, Pnumber, Plocation, Dnumber)
WORKS_ON (Ssn, Pnumber, Hours)
DEPENDENT (Ssn, Dependent_name, Sex, Bdate, Relationship)
```

Fig. 1. Relational schema of the COMPANY database (adopted from [8])

Mapping relations. All entities from an initial entity-relationship diagram exist as relations in the corresponding relational database schema. Relations express a concept similar to ontology classes. Thus, mapping relations into OWL classes is a straightforward process. OWL classes are defined within the *owl:Class* tag.

Mapping attributes. All attributes A_j in a relation R are mapped to corresponding OWL datatype properties P_j. Their domains and ranges are defined within the *owl:DatatypeProperty* tags (the left part of Fig. 2). The domain (*rdfs:domain*) of all those properties is a class C, which corresponds to the relation R. Each property is given the name of the corresponding attribute in addition with the prefix *has* (e.g. the attribute *Lname*, meaning last name, is translated into the datatype property *hasLname*). The range (*rdfs:range*) of each property is the OWL datatype that conforms to the attribute datatype. The database constraint UNIQUE on an attribute results in creating the OWL constraint *maxCardinality=1* on the corresponding property, while NOT NULL implies the cardinality constraint *minCardinality=1*.

Mapping primary keys. A primary key of a relation is an attribute (or a set of attributes) whose value is distinct for each individual tuple. A property created from a primary key attribute should be declared inverse functional. A property P is inverse functional if $P(domainX, rangeZ) = P(domainY, rangeZ)$ implies $domainX = domainY$

i.e. the same range value always denotes a unique instance of the domain [2]. Relation EMPLOYEE has a primary key attribute *Ssn*. The corresponding OWL structure is the inverse functional property *hasSsn* (the right part of Fig. 2). We also state that *minCardinality* for this property is *1* when referring to class *Employee* (the value of a primary key cannot be NULL).

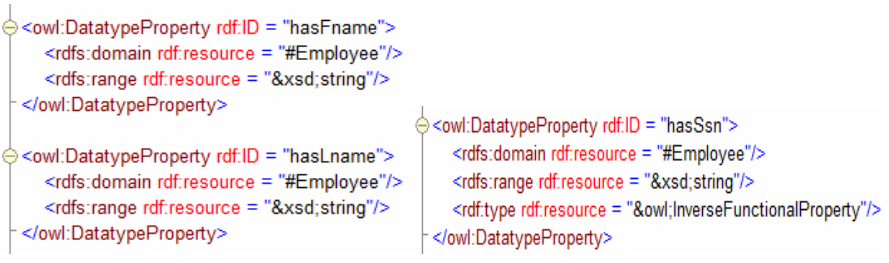

Fig. 2. Mapping attributes and primary keys to an OWL ontology

Mapping binary relationships. Database relations are connected by the mechanism of primary and foreign keys. Hence, if a foreign key attribute in a relation *A* points to the primary key of some other relation *B*, a semantic association exists between them. In the original ER diagram of the database those associations are specified explicitly as bidirectional *binary relationships*, their names and cardinality constraints being declared explicitly as well. The fact that no such information exists in a relational database causes a problem of naming the ontological structures created as the result of the mapping process. The unidirectional primary-foreign key association between relations *A* and *B* is translated into two OWL object properties (*owl:ObjectProperty*) between the corresponding OWL classes *A* and *B*. We introduce a generic naming mechanism, which adds the *haveRelationTo* prefix to the range class of each property.

When a relation *A* points to a relation *B* with its foreign key, the related binary relationship can either have the cardinality 1:1 or N:1. The cardinality is determined by analyzing the database content. The cardinality to-one corresponds to a functional property in OWL (*owl:FunctionalProperty*). The same instance of such a property's domain class must always be joined to the same instance of the range class.

The association between DEPARTMENT.mgr_ssn and EMPLOYEE.ssn has cardinality 1:1. The resulting object properties are presented in Fig. 3. Two functional properties are created, each of them representing one direction of the relationship: *haveRelationToDepartment* and *haveRelationToEmployee*. The domain and range tags inside the object property tag denote the direction of that part of the relationship. The functional property *haveRelationToDepartment* connects each instance of *Employee* (the domain class) to a single instance of *Department* (the range class).

The only difference between mapping 1:1 binary relationships and N:1 relationships to OWL ontologies is the fact that in the latter case only one of the two created object properties is functional: the one that represents the cardinality to-one. The primary-foreign key association between DEPARTMENT.dnumber and EMPLOYEE.dnumber states that an employee works in a single department (the resulting

```
<owl:ObjectProperty rdf:ID = "haveRelationToDepartment">
    <rdfs:comment>foreign key pair (Mgr_ssn/Ssn)</rdfs:comment>
    <rdfs:domain rdf:resource = "#Employee"/>
    <rdfs:range rdf:resource = "#Department"/>
    <rdf:type rdf:resource = "&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID = "haveRelationToEmployee">
    <rdfs:comment>foreign key pair (Mgr_ssn/Ssn)</rdfs:comment>
    <rdfs:domain rdf:resource = "#Department"/>
    <rdfs:range rdf:resource = "#Employee"/>
    <rdf:type rdf:resource = "&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

Fig. 3. Mapping a 1:1 binary relationship to an OWL ontology

```
<owl:ObjectProperty rdf:ID = "haveRelationToDepartment2">
    <rdfs:comment>foreign key pair (dnumber/dnumber)</rdfs:comment>
    <rdfs:domain rdf:resource = "#Employee"/>
    <rdfs:range rdf:resource = "#Department"/>
    <rdf:type rdf:resource = "&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID = "haveRelationToEmployee2">
    <rdfs:comment>foreign key pair (dnumber/dnumber)</rdfs:comment>
    <rdfs:domain rdf:resource = "#Department"/>
    <rdfs:range rdf:resource = "#Employee"/>
</owl:ObjectProperty>
```

Fig. 4. Mapping a N:1 binary relationship to an OWL ontology

functional object property *haveRelationToDepartment2*, see Fig. 4), but more than one employee can work in a department (the non-functional property *haveRelationToEmployee2*).

4 Acquiring Additional Semantics from WordNet

The primary goal of exporting a database into an ontology is to achieve a reliable knowledge source for a particular narrow domain, able to be easily integrated with other ontologies. The integration is impossible without defining a standard reference ontology or dictionary, whose entries have a determined, unambiguous meaning [10]. Thus, we try to associate every class in the target ontology to a WordNet sense.

WordNet [4] is a large taxonomy of the English language, whose searchable lexicon is divided into four categories: nouns, verbs, adjectives and adverbs. Each input word can have more than one meaning (also called word sense). Each word sense can be described by one or more synonyms and is called a synset. A synset is given a description sentence called gloss and may have antonyms.

WordNet includes a set of semantic relations for each word category. The largest spectrum of relations exists for nouns, which comprise about 80% of all WordNet entries [4]. Hyponymy/hypernymy is a transitive relation between nouns that represents subordination/superordination and exactly conforms to the concept of subclasses in ontologies. The part-whole relation is called meronymy/holonymy. Meronymy/holonymy is only occasionally transitive.

The created OWL classes that stem from a database are given some target URI (the value of the *xml:base* attribute, see Fig. 5), while all the associated WordNet-based classes are assigned to the same global URI (*http://www.fer.hr/dbonto/wordnet#*). WordNet class names contain all the synonym words of a synset as well as numbers determining the particular word sense of each word (e.g. *Hall3* in Fig. 5). Classes originating from a database are not declared equivalent to the corresponding WordNet classes (equivalence assumes that two classes are subclasses of each other and share the same set of instances) but only their subclasses (Fig. 5: the database class *Hall* is associated to the third sense of *hall* in WordNet i.e. class *Hall3*). Such a definition

enables equally named classes from other databases (having different properties and thus not equivalent to their namesakes) to be associated to the same WordNet synset. In the ontology integration process those classes will possibly be merged into a single new class whose property set is the union of the two original sets.

```
<rdf:RDF xml:base = "http://www.fer.hr/dbonto/faculty#"
         xmlns:x = "http://www.fer.hr/dbonto/faculty#"
         xmlns:wn = "http://www.fer.hr/dbonto/wordnet#"
    <owl:Class rdf:ID = "Hall">
      <rdfs:subClassOf rdf:resource = "&wn;Hall3" />
    </owl:Class>
</rdf:RDF>
```

Fig. 5. Associating a class resulting from a database to a WordNet synset

Recent word sense disambiguation approaches are based exclusively on dictionary data. They analyze either the semantic similarity between the synsets or their glosses and hyponyms. We use both the similarity calculation technique developed in [14] and the gloss-based technique outlined in [15]. Each table name (future ontology class name) is disambiguated using (1) names of all the table's attributes, (2) names of all tables referenced by the foreign keys in the target table, (3) names of all tables that reference the target table. The disambiguated table names must be WordNet entries (either single-word or multiple-word ones): we can disambiguate table names such as *hall* or *academic_year*, but not *student_course*. On the other hand, names of attributes and other tables may be any combinations of words (in that case the contribution of each word is calculated separately) or abbreviations (the software connects to the Abbreviations.com Web page, www.abbreviations.com, and obtains the meaning).

In [14] WordNet synsets (i.e. word senses) are interpreted as graph vertices, connected by edges representing hyponymy/hypernymy and meronymy/holonymy (each edge is given a weight according to the relation type). All possible paths between the target vertex (corresponding to one sense of the target table name) and all vertices corresponding to different senses of the other word (attribute name, related table name or name particle) are constructed. Weights are multiplied across paths and the highest product becomes the semantic similarity between the target synsets. Since the calculated similarities for different attribute and related table names may point to different word senses, we take the arithmetic mean across all attributes and related tables as the final similarity score. We consider the disambiguation process successful if the highest score is at least 1.2 times bigger than the second highest (the ratio of 1.2 has been obtained by experiments on case study examples).

In [15] word senses are disambiguated by finding the overlap among their sense definitions. For each of the two target synsets the following four sets are extracted: (1) all synonyms, (2) all words of the gloss, (3) synonyms in all hyponyms, (4) all words of all the hyponyms' glosses. The existence of an overlap between any of those four sets belonging to the senses *s* and *s'* of words *w* and *w'*, respectively, suggests a correlation between the senses (i.e. synsets). If no overlap exists for other pairs of senses of *w* and *w'* or if the size of that overlap is smaller (in our case at least 1.2 times), the disambiguation is successful. For example, *w'=computer* disambiguates *w=terminal* since it appears only in the gloss of its third sense.

Experiments are performed for the case study example in Fig. 1 (*company database*) as well as our real-world *faculty database*, which maintains the data about the students and the courses they attend. Disambiguation is needed for about 30% of table

names, which conform to multiple-sense WordNet entries; other 40% of names match single-sense entries. The disambiguation technique presented in [15] shows higher recall, precision and F-measure for both databases than the technique presented in [14] or when both techniques are applied in parallel and only unanimous results considered (Table 1).

Table 1. Comparison of word sense disambiguation techniques

	# of tables	Yang &Powers			Liu, Yu & Meng			both in parallel		
		Rec.	Prec.	F-m.	Rec.	Prec.	F-m.	Rec.	Prec.	F-m.
Company	2	**1.000**	0.500	0.667	**1.000**	**1.000**	**1.000**	0.500	**1.000**	0.667
Faculty	8	0.250	0.000	0.000	**0.875**	**0.857**	**0.867**	0.625	0.800	0.702

5 Prototype Software Tool

A Java prototype tool (Fig. 6) has been developed in order to test our approach on different relational databases as well as to determine which of the presented word sense disambiguation techniques is more suitable for our purpose. The software was encoded in Java 1.5. The JDBC API v3.0 [16] provides access to the input relational databases. The JWNL API v1.3 [17] is used as an interface to WordNet files (we use WordNet v2.1 downloadable from the Princeton website [4]). Jena v2.4 [18] is applied to produce the output OWL ontologies.

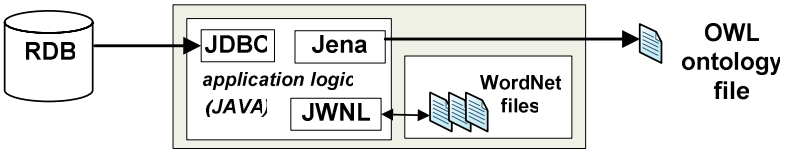

Fig. 6. The architecture of the Java prototype tool

6 Conclusion

This paper presents a framework for an automatic transfer of semantics from relational databases to OWL ontologies in order to exploit the large potential of the "hidden" deep Web relational data in the implementation of the Semantic Web. We use OWL as the target ontology language due to its expressivity and standardization.

In the first phase of the process relational constructs: relations, attributes and primary-foreign key associations between relations, are translated into OWL classes, datatype properties and object properties. The constructed ontologies are enriched in the second phase by acquiring additional semantics from the WordNet lexical database, which defines reference points for integration with other ontologies.

A software implementation of the approach has been developed and tested on two case study examples. The word sense disambiguation mechanism based on analyzing word glosses emerges as the best solution for the second phase of the process.

References

1. World Wide Web Consortium: W3C Semantic Web Activity (last visited: January 16, 2008) (2008), http://www.w3.org/2001/sw/
2. World Wide Web Consortium: OWL Web Ontology Language Guide W3C Recommendation as of February 10 2004 (2004), http://www.w3.org/TR/2004/REC-owl-guide-20040210
3. Bergman, M.K.: The Deep Web: Surfacing Hidden Value. White paper, Deep Content (2001), http://www.brightplanet.com/resources/details/deepweb.html
4. WordNet. A lexical database for the English language (last visited May 5, 2008) (2008), http://wordnet.princeton.edu/
5. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating Data-Intensive Web Sites into the Semantic Web. In: 17th ACM Symposium on Applied Computing, pp. 1100–1107. ACM Press, New York (2002)
6. Astrova, I.: Reverse Engineering of Relational Databases to Ontologies. In: Bussler, C., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 327–341. Springer, Heidelberg (2004)
7. Dogan, G., Islamaj, R.: Importing Relational Databases into the Semantic Web(last visited: January 16, 2008) (2002), http://www.mindswap.org/webai/2002/fall/Importing_20Relational_20Databases_20into_20the_20Semantic_20Web.html
8. Jurić, D., Skočir, Z.: Building OWL Ontologies by Analyzing Relational Database Schema Concepts and WordNet semantic relations. In: Car, Ž., Kušek, M. (eds.) 9th Int. Conf. on Telecommunications, FER, Zagreb, pp. 235–242 (2001)
9. Benslimane, S.M., Benslimane, D., Malki, M.: Acquiring OWL Ontologies from Data-Intensive Web Sites. In: Wolber, D., Calder, N., Brooks, C.H., Ginige, A. (eds.) Int. Conf. on Web Engineering, pp. 361–368. ACM Press, New York (2006)
10. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, New York (2007)
11. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an Algorithm and an Implementation of Semantic Matching. In: Bussler, C., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 327–341. Springer, Heidelberg (2004)
12. Castano, S., Ferrara, A., Montanelli, S.: H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In: Cruz, I.F., Kashyap, V., Decker, S., Eckstein, R. (eds.) Proc. Int. Workshop on Semantic Web & Databases, pp. 231–250 (2003)
13. Pan, R., Ding, Z., Yu, Y., Peng, Y.: A Bayesian Network Approach to Ontology Mapping. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 563–577. Springer, Heidelberg (2005)
14. Yang, D., Powers, D.M.W.: Measuring Semantic Similarity in the Taxonomy of WordNet. In: Estivill-Castro, V. (ed.) 28th Australasian Computer Science Conference, pp. 315–322. Australian Computer Society (2005)
15. Liu, S., Yu, C.T., Meng, W.: Word Sense Disambiguation in Queries. In: Herzog, O., et al. (eds.) 2005 ACM International Conference on Information and Knowledge Management, pp. 525–532. ACM Press, New York (2005)
16. Java Database Connectivity - JDBC (last visited May 5, 2008) (2008), http://java.sun.com/products/jdbc/overview.html
17. Java WordNet Library - JWNL (last visited May 5, 2008) (2008), http://jwordnet.sourceforge.net
18. Jena Semantic Web Framework (last visited May 5, 2008) (2008), http://jena.sourceforge.net