WRAPPING AND INTEGRATING HETEROGENEOUS RELATIONAL DATA WITH OWL

Seksun Suwanmanee , Djamal Benslimane, Pierre-Antoine Champin
LIRIS laboratory, University of Lyon 1
8, Bd Niels Bohr 69622 Villeurbanne cedex, France

Philippe Thiran
Computer Science Department, Eindhoven University of Technology
Den Dolech 2, 5600 MB Eindhoven, The Netherlands

Keywords: Database, Integration, Mediator, Wrapper, Semantic Web, Ontology, OWL

Abstract: The Web-based information systems have been much developed since the Internet is known as a global accessible open network. The Semantic Web vision aims at providing supplementary meaningful information (meta-data) about Web resources in order to facilitate automatic processing by machines and interoperability between different systems. In this paper, we present an approach for the integration of heterogeneous databases in the Semantic Web context using semantic mediation approach based on ontology. The standard OWL language is used here as the ontology description language to formalize ontologies of local data resources and to describe their semantic correspondences in order to construct an integrated information system. We propose an architecture adopting mediator-wrapper approach for a mediation based on OWL. Some illustrations of database wrapping and semantic mediation using OWL are also presented in the paper.

1 INTRODUCTION

Since the explosive expansion of the Internet, the technology in this area has been progressing and the amount of data available on the Web has rapidly increased. Many information systems can expose their data via the internet that facilitates the remote access. The information system that used to work locally can become online accessible and ready to communicate with other systems. In this work, we focus on the semantic integration of data-oriented information systems within the Internet. In such systems, databases are modeled and implemented independently. An adaptable mediation system is therefore necessary to allow cooperation between them. The mediation plays an important role in the Semantic Web context in which information may not be processed from a single data source, but instead from combinations of multiple heterogeneous data sources with different representations of a common domain. Here we propose a mediator-wrapper approach based on OWL ontology in the Semantic Web context for integrating heterogeneous databases.

The Semantic Web began with the idea that the Web resources should also provide meta-data or semantic description about the resources themselves. These meta-data can allow intelligent agents to work with. The W3C first introduced RDF/RDFS (http://www.w3c.org/RDF/) as a Semantic Web language. Then to support the needs of ontology language for the Web resources, OWL (http://www.w3c.org/2004/OWL/) was recently designed based on the RDF graph model and the semantic found of description logics.

Recently, OWL has become the determinant standardization effort of the international research community in this area. This implies that in the future we will see many ontologies in specific knowledge domains expressed in OWL. It is of crucial importance therefore to be able to integrate

Suwanmanee S., Benslimane D., Champin P. and Thiran P. (2005).
WRAPPING AND INTEGRATING HETEROGENEOUS RELATIONAL DATA WITH OWL.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 11-18
DOI: 10.5220/0002527200110018

ontologies in order to provide the interoperability of different independent data sources.

The rest of this paper is organized as follows. In Section 2, we describe main characteristics of OWL. Section 3 presents the general architecture of our approach and describes in detail the database wrapping and the ontology mediation using OWL. An experimental implementation is shown in Section 4. Finally, Section 5 concludes this paper.

2 OWL: WEB ONTOLOGY LANGUAGE

An ontology describes concepts and relations for representing and defining a specific knowledge domain. Essentially, it consists of a hierarchical description of concepts in a domain, along with descriptions of the properties of each concept and maybe instances of concepts.

As mentioned in many works such as (Cruz, 2004), (Horrocks, 2003a), (Mena, 2000), ontology can play an important role in the semantic mediation by providing a source of shared and precisely defined terms that can be used in meta-data.

RDF (Resource Description Framework), and RDF Schema (RDFS) have been widely accepted as a formal language of meta-data describing any Web resources. RDFS in particular is recognizable as an ontology knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (specialization) relations. RDFS has, however, some limitations that cause difficulties for automated reasoning process. A new Web ontology language, DAML+OIL, was developed on top of the RDF model. This work led to OWL (Web Ontology Language), now officially recommended as the ontology language for the Semantic Web by W3C.

OWL uses the same syntax as RDF (and RDFS) to represent ontologies. It may thus appear in several formats such as RDF/XML serialization, N-Triples, N3. It also has a compact abstract syntax which we use in this paper since it is less verbose than pure RDF syntaxes.

Concretely, an OWL ontology consists of definitions and descriptions of concepts (or classes) and relations (or properties) between them. There are basic elements of OWL (some come from RDF/RDFS) that allow to define classes, to describe their hierarchical relations and also their properties. All classes are typed `owl:Class`. The expression `rdfs:subClassOf` decribes an inclusion relation between classes in a hierarchy. `owl:equivalentClass` is used to declared the equivalence of classes.The properties are of two types: `owl:DatatypeProperty` and `owl:ObjectProperty`. A datatype property is a binary relation that associates an individual of a class to a value (or values) of a simple data type defined in accordance with XML Schema datatypes such as integer, string. On the other hand, an object property relates individuals of classes (or of a same class). When a property is defined, we usually specify its domain (`rdfs:domain`) and its range (`rdfs:range`). We can also characterise a property by specifying its supplementary type such as `owl:transitiveProperty`, etc.

OWL is classified into three species: OWL Lite, OWL DL (description logic) and OWL Full. OWL DL which is used in the scope of this work is particularly interesting since it has enough expressivity and a decidable reasoning mechanism (Horrocks, 2003b).

3 APPROACH

In this section, first we present an overview of our approach then show some motivating examples which illustrate the functional aspects of the proposed approach. The details of wrapping part and the ontology integration are described in later subsections.

3.1 General architecture

Our system consists of a collection of data sources and a mediator that facilitates the access to local data and reconciles semantic conflicts among those local systems. Our approach adopts a so-called mediator-wrapper architecture that allows local systems to operate independently while the remote access can be done via a mediator and adaptable wrappers. This mediation system provides a transparent access of different local sources to the user. Figure 1 illustrates the architecture of our approach that is divided into three layers:

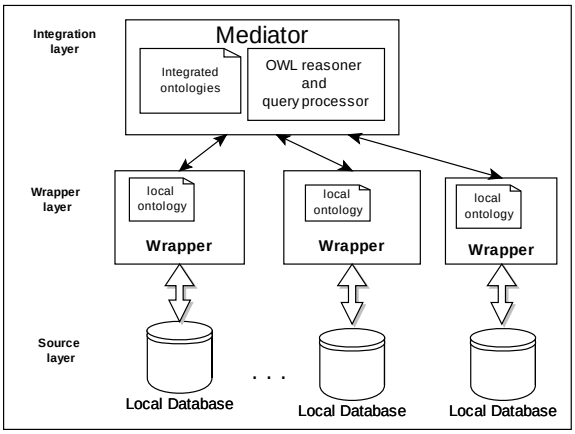

Figure 1: Overview of the general architecture

Source layer contains a set of autonomous databases as local data sources of a common domain. These databases model their data independently according to thier requirements and applications.

Wrapper layer includes wrappers for each local database. These wrappers allow the interface between local systems and the mediator. In the context of Semantic Web, the wrapper provides an OWL ontology representing a data source and a means to access and to query the local source. More details related to the Wrapper layer are given in section 4.

Integration layer contains a mediator which allows the interoperability of the local sources. One of its main functions is to integrate local ontologies in order to facilitate a global access of local sources. Since different local ontologies may present some semantic conflicts, ontology mappings are necessary to overcome these differences. The mediator also contains a reasoning engine that works on the OWL ontologies and the mappings, and a query processor which allows the users to retrieve data from local sources.

The query processing is beyond the scope of this paper. Let us now focus on the ontology mediation based on OWL. The following subsection presents an example of heterogeneous databases and summarizes some semantic conflicts.

3.2 Motivating examples

Two relational databases are presented here as example of local sources. Suppose that *Database A* is a local music database of *SchoolA*. A simplified schema of the database is shown in Figure 2. It contains information about people and courses of the school. The data concerning students and teachers are kept separately in different tables. The column `teaches` of `Teacher` indicates course(s) which a teacher is responsible for. `MusicClass` contains data of all music classes which are classified into categories by their music instrument for example, `PianoClass` and `ViolinClass`.

DATABASE A

```
People(pID, name, age)

Teacher(tID, teaches)
  FK: tID  -> People.pID
  FK: teaches -> MusicClass.cID

Student(sID, attends)
  FK: sID    -> People.pID
  FK: attends -> MusicClass.cID

MusicClass(cID, hours, level)

PianoClass(cID, beginDate)
  FK: cID  -> MusicClass.cID

ViolinClass(cID, beginDate, room)
  FK: cID  -> MusicClass.cID
```

DATABASE B

```
Instructor(instrID, name, officeRoom)

Student(stCode, name, age)

Course(courseID, usesInstrument, hours,
       startDate)
  FK: usesInstrument  ->
        MusicInstrument.instrument

BeginnerCourse(courseID)
  FK: courseID  -> Course.courseID

IntermediateCourse(courseID)
  FK: courseID  -> Course.courseID

AdvancedCourse(courseID)
  FK: courseID  -> Course.courseID

MusicInstrument(instrument)

taughtBy(course, instructor)
 FK: course  -> Course.courseID
 FK: instructor ->Instructor.instrID

hasStudents(course, student)
 FK: course  -> Course.courseID
 FK: student -> Student.stCode
```

Legend
*Table(**PrimaryKey**, columns)*
FK:ForeignKey->TargetTable.column

Figure 2: Relational schemas of the example databases

Another school may model its music database in a different way as shown in Figure 2. In Database B of *SchoolB* the music classes are categorised into 3 tables according to the level (beginner, intermediate and advanced). Table `Instructor` stores data about teachers and `Student` contains those about students. `taughtBy` and `hasStudent` associate a course to its teacher and students respectively.

In our approach, these databases are exposed by means of a wrapper which provides OWL ontologies representing local databases.

3.3 Wrapping databases into OWL

Wrappers are developed on top of each local data sources and provide a standard and common interface to facilitate and homogenize their access. This interface is made up of: (1) a *local ontology* of the wrapped data source, expressed in OWL and (2) a *query language* which uses the semantics defined in the local ontology.

3.3.1 Local Ontology

In order to export the local sources in OWL, we need to define how a source schema expressed in any modeling language can be mapped onto the OWL data model. For this purpose existing works can be used. In (Lehti, 2004) a mapping of XML schemas to OWL is presented. There is also tool such as D2R (bizer, 2003) which propose a flexible mapping language to generate RDF description of relational data that can be easily adapted to OWL format.

3.3.2 Query Language

In our case, queries need to be based on OWL; that means that the query language needs a formally defined semantics for the OWL data model. Therefore one could use and slightly modify OQL or one of the RDF query languages (Fransincar, 2004, Karvounarakis, 2002) because there are also defined on a graph models. Recently, (Lehti, 2004) proposed the query language SWQL which specializes in OWL.

3.3.3 OWL ontologies of the example databases

In the present time, we develop a simple wrapper that provides an OWL ontology corresponding to a given relational schema (see Section 4 for the prototype wrapper). The OWL ontology is generated automatically according to predefined basic translation rules: basically one class per table, one data property per column, one object property per foreign key and also one object property for a table containing only a pair of foreign keys that forms the primary key.

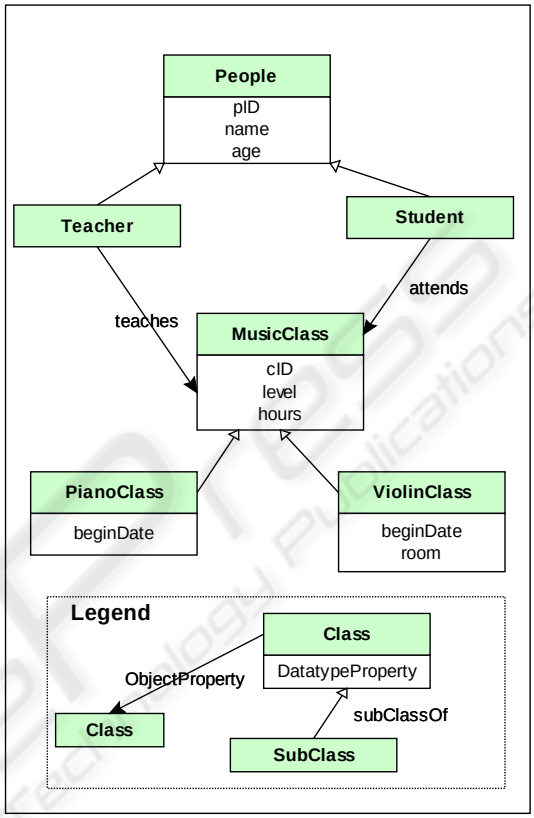

Figure 3: Ontology of the music school *SchoolA*

As an illustration of the generation of ontology by the translation rules, we apply them to the example databases in the previous section. Figure 3 depicts the ontology of the *SchoolA* in a hierarchical diagram of classes. This ontology is the result of the automated application of the translation rules. However, the user may also enrich the generated ontology with additional descriptions. For instance, the inheritance relation between `Teacher` (and `Student`) and `People` is added into the initially generated ontology.

The OWL abstract code below show some parts of the ontology of *SchoolA* in Figure 3. `partial` (`complete`) can be simply read as `subClassOf` (`equivalentClass`, respectively). The rest is self-explained.

```
Ontology ( SchoolA
  Class (MusicClass partial)
  Class (PianoClass partial MusicClass)
  Class (ViolinClass partial MusicClass)
```

```
  Class (People partial)
  Class (Student partial People)
  Class (Teacher partial People)
  ObjectProperty(attends domain(Student)
     range(MusicClass) )
  ObjectProperty(teaches  domain(Teacher)
     range(MusicClass) )
  DataProperty(level  domain(MusicClass)
     range(xsd:string) )
  DatatypeProperty(name  domain(People)
     range(xsd:string) )
  DatatypeProperty(age
    range(xsd:positiveInteger))
. . .
)
```

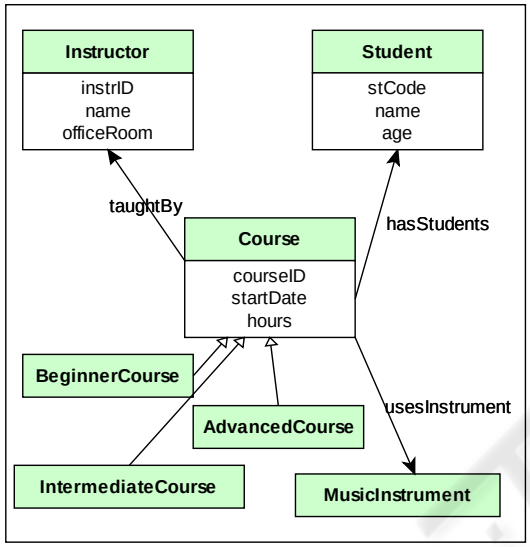

Figure 4: Ontology of the music school B.

Similarly, the OWL ontology *SchoolB* provided by the wrapper is defined from its relational schema. Figure 4 represents graphically the ontology and the corresponding OWL code is described as follows:

```
Ontology ( SchoolB
  Class (Course partial)
  Class (BeginnerCourse partial Course)
  Class (IntermediateCourse partial
           Course)
  Class (AdvancedCourse partial Course)
  Class (Student partial People)
  Class (Instructor partial People)
  ObjectProperty(taugtBy
    domain(MusicCoruse) range(Instructor))
  ObjectProperty(hasStudent
    domain(MusicCoruse) range(Student))
  ObjectProperty(usesIntstrument
    domain(MusicCourse)
    range(MusicInstument) )
  DatatypeProperty(name
    domain(People) range(xsd:string))
  . . .
)
```

These two local ontologies present differences in several aspects. In addition to the different terminology (e.g. teacher/instructor, music class/course, etc), the classification of music classes uses different criteria. There are also some differences in properties such as in *SchoolA* the property `teaches` relates a music teacher to a class (or classes) whereas the property `taughtBy` in *SchoolB* does in the inverse direction. These two properties are an inverse of each other.

In a general context, we have a set of independent local data sources of a common domain and we need to share and exchange information among them. Each data source can be represented by its proper ontology that uses a certain vocabulary with specific semantics behind. An appropriate mediation system is needed for allowing the interoperability of different data sources. This mediation system must provide a means to overcome the semantic heterogeneity between the local systems and also a means to access to local information with transparency as much as possible. In the next section we describe a way of integration our approach for ontology integration based on the formalization in OWL.

3.4 Integrating ontologies

The ontology integration in our approach consists of mappings of elements of different OWL ontologies. OWL provides sufficient elements for expressing relations between classes and between properties as well. Moreover, these expressions are not limited in a same ontology. As a result, we can apply OWL to describe the mappings of different ontologies. Our objective is to obtain an integrated ontology which contains semantic mappings of different local ontologies.

We illustrate how to use OWL as ontology mapping language by showing the mapping between previous motivating examples.

3.4.1 Ontology importing

In the mediator level, it is important to specify the predefined involved ontologies by their URI so that the rest of ontology description can refer to the existing elements that are previously defined in local ontologies. This reference is described by the OWL expression `owl:import`. OWL abstract code of importing our predefined example ontologies *SchoolA* and *SchoolB* is shown as follows:

```
Ontology ( IntegratedAB
  Annotation (owl:imports
       "http://music.school/schoolA")
  Annotation (owl:imports
      "http://music.school/schoolB")
  ...
)
```

3.4.2 Class mapping

Basic relations of classes such as inclusion, equivalence and disjunction allow us not only to describe a hierarchical structure of classes in one ontology but we can also apply these class relations to establish mappings between classes from different ontologies. The inclusion (expressed by `rdfs:subClassOf`) represents the subsumption between two classes (parent class subsumes subclass). The class equivalence relation of two classes (`owl:equivalentClass`) implies that the inclusion holds in the two directions. On the other hand, two classes which have no individual in common can be declared mutually disjoint (`owl:disjointWith`).

Here are examples of simple class mappings which describe some corresponding concepts in the ontologies *SchoolA* and *SchoolB*. In OWL abstract syntax `complete` represents `equivalentClass`.

```
Namespace(schoolA =
   http://music.school/schoolA#
)Namespace(schoolB =
   http://music.school/schoolB# )
Ontology (IntegratedAB
  ...
  Class(schoolB:Student complete
           schoolA:Student)
  Class(schoolB:Instructor complete
           schoolA:Teacher
  Class(schoolB:Course complete
           schoolA:MusicClass)
  ...
)
```

OWL provides some expressions to construct a concept that represents a class of individuals which satisfy some common conditions. A complex class can also be formed by classical set operations like union, intersection and complement. The restrictions and complex class constructions allow us to describe complicated and precise classes. In OWL we can specify a restriction on certain property according to its associated value (`owl:hasValue`), its range of values (`owl:someValuesFrom` and `owl:allValuesFrom` for existential and universal condition respectively) and its cardinality (`owl:min/max/Cardinality`). Besides, OWL provides built-in construct for the usual set operations such as union and intersection to form a more complex class.

For instance, we can describe that the music classes of *SchoolA* that have the advanced level are considered as members of `AdvancedCourse` class of *SchoolB*. This mapping rule can be formulated in OWL as follows.

```
Class(schoolB:AdvancedCourse complete
 restriction(schoolA:MusicClass
    hasValue ("advanced") )
```

In the case of violin class, the music courses of *SchoolB* that use either violin or cello can be considered as a member of `ViolinClass` of *SchoolA*, we may map `ViolinClass` to the union of two restricted music courses of *SchoolB* as follows.

```
Class(schoolA:ViolinClass complete
  unionOf(
    restriction(schoolB:useInstrument
              hasValue (schoolB:Violin))
     restriction(schoolB:useInstrument
              hasValue (schoolB:Cello) )) )
```

3.4.3 Property mapping

We determine a relation between two properties by comparing their members. Three possible relations of properties are inclusion, equivalence and inverse. Property *P1* is a subproperty of *P2* (`rdfs:subPropertyOf`) means that if *P1(x,y)* holds then *P2(x,y)* holds. *P1* and *P2* are equivalent properties (`owl:equivalentProperty`) when *P1(x,y)* if and only if *P2(x,y)*. *P1* is an inverse (`owl:inverseOf`) of *P2* when *P1(x,y)* if and only if *P2(y,x)*.

In our examples, some properties such as `schoolA:name` and `schoolB:name` , `schoolA:beginDate` and `schoolB:startDate` can be mapped as equivalent properties. On the other hand, the property `schoolA:teach` is exactly the reverse of property `schoolB:taughtBy` because a teacher *X* who teaches a class *Y* implies that the class *Y* is taught by the person *X* and the vice visa holds too. Here are some examples of property mappings:

```
EquivalentProperties(
   schoolA:name    schoolB:name)
EquivalentProperties(
   schoolA:hours    schoolB:hours)
ObjectProperty(schoolA:attends
    inverseOf(schoolB:hasStudents) )
ObjectProperty(schoolA:teaches
    inverseOf(schoolB:taughtBy))  . . .
```

4 EXPERIMENTAL IMPLEMENTION

As in an early step of our work, we experiment our approach with some database and ontology examples. An automatic OWL ontology generator that functions over relational databases is implemented as the wrapper layer in our architecture. The mediator is simulated by an OWL inference engine that provides a means to query over integrated ontologies.

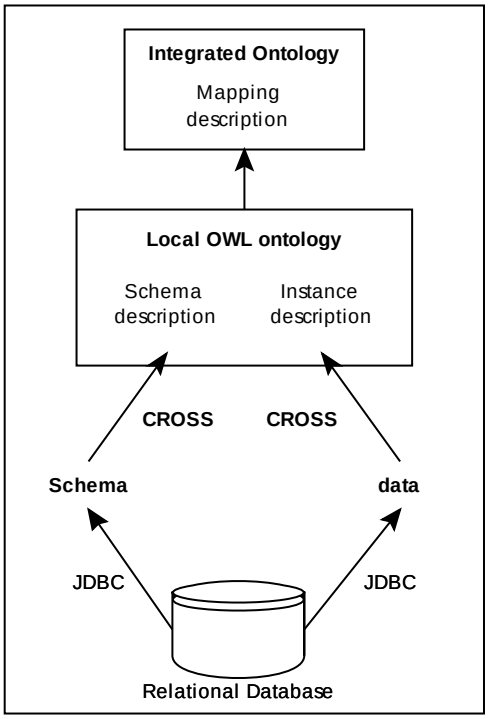

Figure 5: Ontology integration with CROSS

Our wrapper (called CROSS) is developed for exposing relational databases (schema and content) to OWL. This wrapper is written in Java and can access any RDBMS providing a JDBC interface (We only tested it with MySQL and PostgreSQL for the moment). The originality of CROSS compared to existing tools and approaches lies in the fact that it offers a direct bridge from the relational model to OWL (without an intermediate XML step), handles both the schema and the data unlike D2R (Bizer, 2003), and do not rely on any new language to express the mapping between the relational data and the OWL description. CROSS provides a fully automated translation of the relational schema into a first OWL ontology based on the basic principles briefly described in Section 3.3.3. This ontology can then be enriched manually and also import other ontologies.

For integrating generated OWL ontologies, mappings rules need to be defined by the user. All mappings are expressed in OWL that allows an OWL reasoner to work with all concerned ontologies. Figure 5 depicts an overview of the functioning of the prototype.

In our experiment, we use Racer system (Haarslev, 2001) as OWL reasoner which is one of powerful description logic reasoners publicly available. OWL compatibility is a new feature of Racer that allows us to use it as an ontology reasoning engine. We can load an OWL ontology into Racer system using the particular interface program called RICE and it can verify a consistency of the ontology and display a general classification of all concepts defined in the underlying ontology.

By loading an integrated ontology containing ontology imports and description of ontology mappings, RICE program shows a global hierarchy of all classes from different ontologies. We can select a particular class and see all instances of the selected class. Besides, RICE provides an interactive querying system that allows us to make queries over loaded ontologies to the running Racer server. However these queries are formed in the Racer syntax.

We can formulate our query in OWL by a class definition in the integrated ontology. A query can be described by using any terms of imported ontologies and the result of the query comes from all involved local ontologies.

Here are some examples of query expressed in OWL:

```
Class(Q_PianoTeacher complete
  restriction(schoolA:teaches
    someValuesFrom(schoolA:PianoClass)) )

Class(Q_nonEmptyClass complete
    restriction(schoolB:hasStudents
        minCardinality(1))  )

Class(Q_advancedViolinClass complete
intersectionOf(
        schoolA:ViolinClass
        schoolB:AdvancedCourse) )
```

The first class contains all teachers who teach at least one piano class. According to the well-defined mappings between *SchoolA* and *SchoolB*, the result are all piano teachers from the two ontologies. **Q_nonEmptyClass** includes all music classes of *SchoolA* and *SchoolB* that are not empty. This means that they must contain at least 1 student.

The last example query describes a class of the intersection of `schoolA:PianoClass` and `schoolB:AdvancedClass`. Therefore this class includes all violin classes of the advanced level from the two ontologies.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed an approach based on OWL for semantic integration of heterogeneous databases in the context of Semantic Web. We described our mediator-wrapper architecture and the

ontology mediation with OWL. Then we showed some experiments on OWL ontology integration.

Many open issues are not discussed in this paper such as instances in ontology, query processing, to mention a few. These are subjects for future work.

It is also interesting that we move toward an open distributed system which is suitable for the Web context, especially, the P2P architecture. Several approaches have been proposed in the literature for this particular decentralized system (Löser, 2003), (Halevy, 2003). The use of a distributed ontology is also an interesting problem and constitutes an open issue (Goasdoué, 2003). At last but not least, the wrapper CROSS proposed can be improved by adding more options for a better and flexible ontology generation from database schemas.

REFERENCES

Bizer, C. D2R map - a database to RDF mapping language. In *Proc of12th Intl World Wide Web Conf*, Budapest, May 2003. URL:http://www.wiwiss.fuberlin.de/suhl/bizer/d2rmap/D2Rmap.htm

Cruz I. F., Xiao H., and Hsu F. An ontology-based framework for XML semantic integration. In *Proc. of IDEAS* 2004. IEEE Computer Society, 2004.

Frasincar F., Houben G.J., Vdovjak R., Barna P., RAL: An Algebra for Querying RDF, *World Wide Web, Internet and Web Information Systems (WWW)*, 7(1), p.83-109, 2004, Kluwer Academic Publishers.

Goasdoué F. and Rousset M-C. Querying distributed data through distributed ontologies : a simple but scalable approach. *IEEE Intelligent Systems*,18(5),2003, pp. 60-65.

Haarslev V. and Moller R.. RACER system description. In *Proc. of Int. Joint Conference on Automated reasoning (IJCAR'2001)*, June 18-23, 2001, Siena, Italy, LNCS. Springer-Verlag, Berlin, 2001. URL: http://www.sts.tu-harburg.de/~r.f.moeller/racer/

Halevy A, Ives Z.G., Mork P., Tatarinov I. Peer Data Management Systems: Infrastructure for the Semantic Web. In *Proc of WWW Conference*, 2003.

Horrocks I. and Patel-Schneider P.F. Three Theses of Representation in the Semantic Web. WWW 2003.

Horrocks I. and Patel-Schneider P.F. Reducing OWL entailment to description logic satisfiability. In *Proc.of the 2003 Description Logic Workshop (DL 2003)*, volume 81 of CEUR (http://ceur-ws.org/), pages 1-8, 2003.

Karvounarakis G., Alexaki S., Christophides V., Plexousakis D. and Scholl M., RQL: A Declarative Query Language for RDF, *World Wide Web, Internet and Web Information Systems (WWW)*, 2002, Kluwer Academic Publishers.

Lehti P. and Frankhauser P., "XML Data Integration with OWL: Experiences and Challanges", *4th International Symposium on Applications and the Internet (SAINT'04)*, IEEE CS, 2004

Löser A., Siberski W., Wolpers M., Nejdl W. Information Integration in Schema-Based Peer-To-Peer Networks. In *Proc. of CAiSE*, 2003. 258-272

Mena E., Illarramendi A., Kashyap V., and Sheth A. P. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

Vdovjak R. and Houben G.J. Rdf-based architecture for semantic integration of heterogeneous information sources. In *Proc. of International Workshop on Information Integration on the Web*, Apr 2001.