Database-to-Ontology Mapping Generation for Semantic Interoperability

Raji Ghawi
Laboratory LE2I
University of Burgundy
21000 Dijon, France
raji.ghawi@u-bourgogne.fr

Nadine Cullot
Laboratory LE2I
University of Burgundy
21000 Dijon, France
nadine.cullot@u-bourgogne.fr

ABSTRACT

In order to achieve efficient interoperability of information systems, ontologies play an important role in resolving semantic heterogeneity. We propose a general interoperability architecture that uses ontologies for explicit description of the semantics of information sources, and web services to facilitate the communication between the different components of the architecture. It consists of 1) data provider services for mapping information sources to local source ontologies, 2) a knowledge base for representing reference domain ontology, and 3) several web services for encapsulating the different functionalities of the architecture. In this paper, we focus on a component of the architecture which is a tool, called DB2OWL, that automatically generates ontologies from database schemas as well as mappings that relate the ontologies to the information sources. The mapping process starts by detecting particular cases for conceptual elements in the database and accordingly converts database components to the corresponding ontology components. A prototype of DB2OWL tool is implemented to create OWL ontology from relational database.

1. INTRODUCTION AND MOTIVATION

In order to achieve an efficient interoperability between heterogeneous information systems, many solutions have been proposed. Particularly, ontologies play an important role in resolving semantic heterogeneity by providing a shared comprehension of a given domain of interest. An ontology formally defines different concepts of a domain and relationships between these concepts. Wache et al. in [17] give an excellent survey on ontology-based information integration systems. According to the way of exploiting ontologies in information integration, they distinguish three main ontology based approaches: single, multiple and hybrid. Single ontology approaches use one global ontology to which all information sources are linked by relations expressed via mappings that identify the correspondence between each information source and the ontology. In multiple ontologies approaches, each information source is described by its own ontology and inter-ontology mappings are used to express the relationships between the ontologies. The hybrid approaches combine the two previous approaches. Each information source has its own ontology and the semantic of the domain of interest as a whole is described by a global reference ontology. In these approaches there are two types of mappings: mappings between an information source and its local ontology and mappings between local ontologies and the global ontology.

In addition to ontologies, web services are increasingly used to support the interoperability between different applications and clients over the web using recently developed internet oriented data models, standard and protocols such SOAP, WSDL, XML etc.. Web services guarantee the independence of an application from any particular platform or implementation. We propose a cooperation architecture that uses ontologies to represent the semantic of information sources and web services to facilitate the communication between its different parts. Our architecture belongs to the hybrid ontology approach, using a local ontology for each information source and a global ontology as a reference for the local ontologies. The advantage of wrapping each information source to a local ontology is to allow the development of source ontology independently of other sources or ontologies. Hence, the integration task can be simplified and the addition and removal of sources can be easily supported. Most of the architecture components are encapsulated in web services aimed at performing specific tasks, like mapping, querying and visualization web services.

In our architecture, information sources may contain different types of data structures: data may be structured as databases, semi-structured as XML documents, and/or non-structured as web pages or other type of documents. However, all of these sources must be mapped to a local ontology which will express the semantic of information sources. In this paper, we focus only on the mapping between databases and the local ontology.

Currently there are many approaches and tools to deal with database to ontology mapping. They can be classified into two main categories: approaches for creating a new ontology from a database and approaches for mapping a database to an already existing ontology. For our architecture, we suppose that the local ontology does not exist and may be created from the information source. We have developed a tool called DB2OWL to create ontology from a relational database. It looks for some particular cases of database tables and according to them it determines

VLDB '07, September 23-28, 2007, Vienna, Austria.

which ontology component is created from which database component. This tool also generates a mapping document that preserves the set of transformations between the database and ontology's components performed during the creation phase. This paper is organized as follows. In section 2, we present different database to ontology mapping approaches. We give in section 3 an overview of a general cooperation architecture. In section 4, we introduce the DB2OWL tool, list some notations, illustrate the several particular table cases and present the mapping process. Section 5 presents some related works and concludes with some remarks and future work.

2. BACKGROUND

Mapping is a critical operation in many application domains, such as semantic web, schema or ontology integration, data integration, data warehouses, e-commerce, etc. We can distinguish three types of mapping : 1) schema mapping, 2) ontology mapping, and 3) database-to-ontology mapping, on which we focus in this paper.

1. Schema Mapping : Mappings are established between the schema of the individual databases. This process takes two schemas as input and produces a mapping between elements of the two schemas that correspond to each other. Some interesting works in this area are the works of Fuxman et al. [7] and Miller et al. [10]. We refer also to [14] as a survey on existing approaches.
2. Ontology Mapping : The main purpose of this process is to relate the vocabulary of two ontologies that share the same domain of discourse. Ontology mapping is somewhat similar to database schema matching, but it has many particularities due to the structural and conceptual differences between ontologies and databases. Kalfoglou et al. gives in [8] an excellent survey on ontology mapping.
3. Database-to-Ontology Mapping : This is the process whereby a database and an ontology are semantically related at a conceptual level, i.e. correspondences are established between the database components and the ontology components.

The database-to-ontology approaches may be classified into two main categories as follows.

2.1 Creating an ontology from a database

These approaches create an ontology model from a relational database model and migrates the contents of the database to the generated ontology. The mappings here are simply the correspondences between each created ontological component (concept, property, etc.) and its original database component (table, column, etc.). In these approaches, the database model and the generated ontology are very similar. Mappings are quite direct and complex mapping situations do not usually appear. The creation of ontology structure may be straightforward, involving direct transformations of database tables to ontology concepts and columns into properties. This type of direct mapping is not sufficient for expressing the full semantics of the database domain. The creation of ontology structure may require the discovery of additional semantic relations between database components (like the referential constraints) and take them into account while constructing ontology concepts and relations between them.

2.2 Mapping a database to an existing ontology

In these approaches, it is considered that an ontology and a legacy database already exist. The goal is to create mapping between them, and/or populate the ontology by the database contents. Mappings here are more complex than those in the previous case because different levels of overlap between the database domain and the ontology's one can be found, and those domains do not always coincide because the modeling criteria used for designing databases are different from those used for designing ontology models [2].

Both mapping approaches above include two processes: (1) mapping definition i.e. the transformation of database schema into ontology structure, and (2) data migration i.e. the migration of database contents into ontology instances. The migration of database instances into ontological instances (individuals), also called ontology population, may be done in two ways [13] : either as a batch process by dumping all the database instances to the ontology repository, or as a query driven process by transforming only the database instances that are the response to a given query, i.e. only the data needed to answer the user's query are retrieved from the sources.

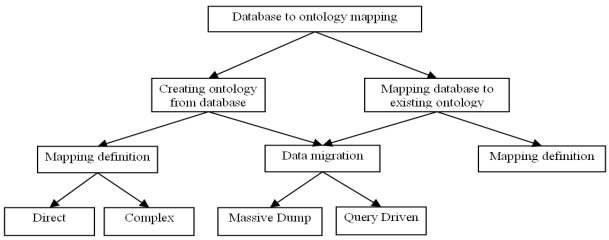

Figure 1. Classification of database-to-ontology mapping approaches.

3. GENERAL ARCHITECTURE

In this section, we give an overview of our interoperability architecture and its main components. It is a cooperation system between several information sources and is aimed at answering user queries on these sources. User queries are submitted only on the reference ontology using the query web service. Thus, users can query heterogeneous and distributed information sources simultaneously and combine the obtained results in order to get information that may not be available directly, i.e. the user has the illusion that he queries a unique source. In order to bridge the gap of heterogeneity between information sources, ontologies are used to describe the semantics of the information sources and to make their contents explicit. The ontologies have to be linked to actual information in order to support the integration process. This is done via mappings between each information source and its ontology. For each incorporated information source, a local ontology is generated to describe its semantics as well as the resulting mappings between the source and the local ontology. Then the local ontologies are mapped to a global ontology using the mapping web service. The global ontology describes the semantics of the whole domain of interest. User's queries are submitted to the query web service that analyses the queries, decompose them into sub-queries which are redelivered to the

relevant data provider services. As shown in figure 2, the cooperation architecture consists of the following components.

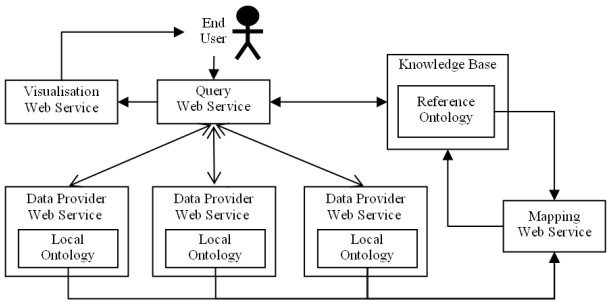

Figure 2. Global architecture of our cooperation system.

3.1 Knowledge Base

This is the main component of the architecture. It contains the reference ontology, a mapping directory, and a toolbox. The reference ontology describes a specified knowledge domain. It represents the global model for local ontology models and is supposed to cover all the local domains, i.e. each concept, role and attribute in any local ontology has a corresponding concept, role and attribute in the reference ontology. The mapping directory contains information about the mappings between the reference ontology and the local ones. The mapping itself is stored in the data provider service. The directory only associates each concept in the referential ontology with a list of local ontologies which are linked to this concept. The toolbox contains tools that are used by the mapping web service to estimate the similarity between ontologies components.

3.2 Data Provider Service

It encapsulates an information source incorporated in the cooperation system. In addition to the information source, the data provider service contains a local ontology representing the semantics of the information source, as well as two types of mappings: information source to local ontology mapping, and local ontology to reference ontology mapping, as described in figure 3. In this paper, we will only deal with information source based on relational databases. The automatic mapping of other models to ontology is beyond the scope of this paper.

The local ontology is automatically generated from the database using the DB2OWL tool which will be presented in details in the rest of this paper. This tool also generates a description of the mapping between database and the resulting local ontology. Our objective is to keep the instances separated from the structure of their ontology. Thus, the generated ontology will contain only the classes and properties but not the instances, which will stay at the database and be retrieved and translated as needed in response to user queries. A data provider service also plays the role of wrapper that translates queries over its local ontology into SQL queries over its data source and reformulates the results in terms of the local ontology.

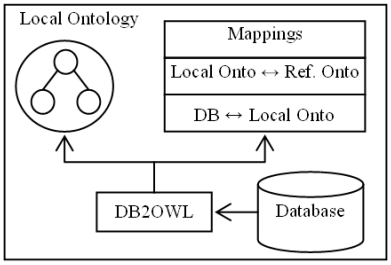

Figure 3. The architecture of the data provider service.

3.3 Mapping Service

This service is used to connect the local ontology to the reference ontology. It compares the two ontologies using the methods defined in the knowledge base toolbox, and produces inter-ontology mappings which will be stored in the appropriate data provider service, as well as an up-to-date version of the mappings directory in the knowledge base.

In general, a mapping web service estimates similarity between the components (concepts and roles) of two ontologies, using structural and semantic (graph based and information value based) methods. The similarity estimating process is out of the scope of this paper.

3.4 Query Service

When a query is submitted to the system, it is analyzed by this service and decomposed into a set of modular queries. Then using the mapping directory in the knowledge base, the query web service redirects the single queries to the suitable data provider services.

In fact, queries are expressed in SPARQL language [12], therefore, a query is composed of a set of triple patterns. Each triple pattern corresponds to a concept or a property in the reference ontology. For each local ontology, a sub-query is established by selecting from the global query the triple patterns that are relevant to this local ontology (according to the mapping directory). Each sub-query is then redelivered to the appropriate data provider service. In other words, each data provider service will receive only a subset of query triple patterns which are covered by its local ontology.

When an SPARQL query is received by a data provider service, it is translated to an SQL query using the mappings between the database and the local ontology. The SQL query is executed in the database and its result is encapsulated as an SPARQL response and returned to the query web service. The query web service then collects the responses returned from data provider services and recomposes them in one coherent response which will be sent to the visualization web service.

3.5 Visualization Service

The final response will be redirected to the visualization web service which is responsible for presenting the query result in a suitable way. The visualization process is out of the scope of this paper. The following section introduces our approach DB2OWL as well as the mapping process which it uses.

4. DB2OWL MODULE

The goal of the DB2OWL module is to automatically create a new ontology from a relational database. In our architecture, DB2OWL is exploited by the data provider service to generate a local ontology for each data source (relational database). Currently, DB2OWL is not intended to map several databases to one ontology. However, reconciling different data sources is performed by mapping their local ontologies to the reference ontology. This task is carried out by the mapping web service.

The created ontology is described in OWL-DL language[1], a W3C recommendation for publishing and sharing ontologies on the web. OWL-DL is based on Description Logics [1], a family of knowledge representation languages, which is characterized by its expressiveness and reasoning power. The mapping process starts by detecting some particular cases for tables in the database schema. According to these cases, each database component (table, column, constraint) is then converted to a corresponding ontology component (class, property, relation). The set of correspondences between database components and ontology components is conserved as the mapping result to be used later. In the following subsections, we introduce the notation used to describe the database metadata and explain the table cases which must be detected in the database in order to exploit them throughout our mapping process. The mapping process itself is then introduced, and finally we illustrate the mechanism of mapping generation during the process.

4.1 Notations

Let DB be a database and let T be a table of DB, we note col(T), P(T) and F(T) the sets of columns, primary keys and foreign keys of table T respectively. We note also PF(T), P_(T), _F(T), __(T) the set of columns which are respectively both primary and foreign keys, primary but not foreign keys, foreign but not primary keys, and not primary nor foreign keys. The sets PF(T), P_(T), _F(T), __(T) are a partition of col(T).

A referential integrity constraint is represented by the quadruplet ric (T_1, A_1, T_2, A_2) where T_1, T_2 are tables and $A_1 \subseteq col(T_1)$, $A_2 \subseteq col(T_2)$ are set of columns of the tables T1 and T2 such that each element of A_1 is a foreign key referenced by an element of A_2 i.e. $\forall \alpha_i \in A_1, \exists \beta_i \in A_2$, α_i is referenced by β_i , so $A_1 \subseteq F(T_1)$ and $A_2 \subseteq P(T_2)$.

Let $\mathcal{RIC}$ be the set of all explicit referential constraints in a DB, we define additional functions for a referential integrity constraint: the local table (LT) and the local attributes (LA) functions which respectively give the reference table (the owner) and attributes of the constraint. The referenced table (RT) and referenced attributes (RA) functions that respectively give the table and the attributes referenced by the constraint. So LT(ric) = T_1 , LA(ric) = A_1 , RT(ric) = T_2 , and RA(ric) = A_2 . For a table T, we also define the function RIC(T) which returns the set of referential integrities whose local table is T, i.e. RIC: DB $\rightarrow \mathcal{P}(\mathcal{RIC})$, RIC(T) = { ric(T_1, A_1, T_2, A_2) $\in$ RIC , LT(ric) = T}.

4.2 Different table cases

The mapping process used in our approach depends on particular database table cases that are taken in account during the ontology creation. These cases will be illustrated using the following example database, which represents a library database.

AUTHOR (<u>authorNo</u>, name)
REFERENCE (<u>refNo</u>, title, year)
PUBLISHER (<u>publisherNo</u>, pubName, pubAddress, pubTelNo)
BOOK (<u>refNo</u>, ISBN, publisherNo)
JOURNALARTICLE (<u>refNo</u>, journal)
REFAUTHOR (<u>refNo</u>, <u>authorNo</u>)
REFCOPY(<u>catalogNo</u>, shelf, refNo, dateInStock)

4.2.1 Case 1

When a table T is used only to relate two other tables T_1, T_2 in a many-to-many relationship, it can be divided into two disjoint subsets of columns A_1, A_2, each participating in a referential constraint with T_1 and T_2 respectively:

RIC(T) = {ric_1, ric_2} : ric_1 (T, A_1, T_1, $P(T_1)$), ric_2 (T, A_2, T_2, $P(T_2)$)

Therefore all T columns are foreign keys and they are primaries as well because their combination uniquely defines the rows of T, i.e. col(T) = F(T) = P(T), so: col(T) = PF(T) .

Thus, the necessary and sufficient condition for a table T to be in case 1 is: col(T) = PF(T) and |RIC(T)| = 2 .

Example: Let us consider the table « REFAUTHOR » that consists of two columns {refNo , authorNo }. We note that P(T) = {refNo , authorNo } and F(T) = {refNo , authorNo }, so PF(T) = {refNo , authorNo } = col(T). In addition, RIC(T) = {ric1, ric2} where: ric1 (REFAUTHOR, {refN}, REFERENCE {refNo}) and ric2 (REFAUTHOR, {authorNo} , AUTHOR {authorNo}) so |RIC(T)| = 2, therefore REFAUTHOR is in case 1.

4.2.2 Case 2

This case occurs when a table T is related to another table T_1 by a referential integrity constraint whose local attributes are also primary keys, i.e. $\exists$ ric $\in$ RIC(T), LA(ric) = P(T) , in other words: ric (T, P(T), T_1, $P(T_1)$). In this case all the primary keys of T are foreign keys because they participate in a referential integrity constraint: P_(T) = $\varnothing$.

Thus, the necessary and sufficient condition for a table T to be in case 2 is: $\exists$ ric $\in$ RIC(T) , LA(ric) = P(T) .

Example: Let us consider the table «BOOK» consisting of the columns {refNo, ISBN, publisherNo}. We find that P(T) = {refNo} and RIC(T) = {ric1, ric2} where: ric1 (BOOK, {refNo}, REFERENCE {refNo}), and ric2 (BOOK, {publisherNo}, PUBLISHER {publisherNo}). We note that LA(ric1) = {refNo} = P(REFERENCE) , therefore BOOK is in case 2

4.2.3 Case 3

This case is the default case, it occurs when none of previous cases occur.

Example: Let us consider the table «AUTHOR», it consists of the columns: {authorNo, name}. We note that P(T) = {authorNo} and F(T) = {}, so PF(T) = {}. At the other hand, RIC(T) = {} , so MODULE is not in case1 nor in case2, therefore it is in case3.

The different cases are summarized in Table 1.

[1] http://www.w3.org/TR/owl-features/.

Table 1. The different particular cases used in mapping process.

Case	Necessary and sufficient condition	Example
case1	col(T) = PF(T) and |RIC(T)| = 2	REFAUTHOR
case2	$\exists$ ric $\in$ RIC(T), LA(ric) = P(T)	BOOK
case3	T is not in case1 nor in case2	REFCOPY

When these different cases are detected in the database, the mapping process can use them to appropriately map database components to suitable ontology components as follows.

4.3 Mapping Process

The mapping process is done progressively as follows. It starts by mapping the tables to concepts and then mapping the columns to properties. Thus, the table cases mentioned above are used twice: one time for table-to-class mapping and the other time for column-to-property mapping. The mapping process consists therefore of the following steps:

1. The database tables that are in case 3 are mapped to OWL classes.
2. The tables in case 2 are mapped to subclasses of those classes corresponding to their related tables, i.e. if T is in case 2 then there is a referential integrity constraint ric $\in$ RIC(T) where ric (T, P(T), T_1, P(T_1)), so T is mapped to a subclass of the class corresponding to T_1 . For example, the tables BOOK and JOURNALARTICLE are mapped to subclasses of the class corresponding to the table REFERENCE.
3. Each table in case 1 is not mapped to class, but the many-to-many relationship that it represents is expressed by object properties. Two object properties are added, one for each class whose corresponding table was related to the current table. In other words, when a table T is in case 1 then there are two referential constraints: ric_1 (T, A_1, T_1, P(T_1)) and ric_2 (T, A_2, T_2, P(T_2)), and if c_1, c_2 are the two classes corresponding to T_1, T_2 respectively, so we assign to c_1 an object property op_1 whose range is c_2 , and assign to c_2 an object property op_2 whose range is c_1. Each of these two properties op_1, op_2 are inverse to the other. For example, the table REFAUTHOR is in case 1, it relates two other tables REFERENCE and AUTHOR, so it is not mapped to a class, but we assign to the class AUTHOR an object property REFAUTHOR.refNo whose range is the class REFERENCE, and we assign to the class REFERENCE an object property REFAUTHOR.authorNo whose range is the class AUTHOR.
4. For tables that are in case 3, we map their referential constraints to object properties whose ranges are classes corresponding to their related tables; i.e. if a table T is in case 3 and has a ric(T, A, T_1, A_1) and if c, c_1 are the classes corresponding to T, T_1 respectively, then we assign to c an object property op whose range is c_1, and we assign to c_1 an object property op' whose range is c. To preserve the original direction of the referential constraint from T to T_1, we set the object property op as functional. So it will have at most one value for the same instance. This characteristic is obvious because it comes from the uniqueness of key. For example, the table REFCOPY is in case 3 and it has a referential integrity constraint with the table REFERENCE, so we assign to its corresponding class an object property REFCOPY.refNo which is functional and whose range is the class corresponding to table REFERENCE, and we assign to the class corresponding to REFERENCE an object property REFERENCE.REFCOPY whose range is the class corresponding to REFCOPY, each of those object properties is inverse to the other.
5. For tables that are in case 2 and have other referential constraints than the one used to create the subclass, we map them to object properties as in the previous step. For example, the table BOOK is in case 2 and has a referential integrity constraint with the table PUBLISHER (other than its constraint with REFERENCE which is used to make it a subclass), so we assign to BOOK an object property BOOK.publisherNo which is functional and whose range is PUBLISHER, and we assign to PUBLISHER an object property PUBLISHER.BOOK whose range is BOOK.
6. Finally, for all tables we map their columns that are not foreign keys to datatype properties. The range of a datatype property is the XML schema data type [3] equivalent to the data type of its original column. The column NAME in the table AUTHOR is mapped to a datatype property AUTHOR.NAME whose range is XSD string datatype.

An example of resulting OWL ontology is given in figure 5.

```
<owl:Class rdf:ID="&BOOK">
  <rdfs:subClassOf rdf:resource="REFERENCE"/>
</owl:Class>
<owl:FunctionalProperty rdf:ID="BOOK.PUBLISHERNO">
  <rdfs:domain rdf:resource="#BOOK"/>
  <rdfs:range rdf:resource="#PUBLISHER"/>
</owl:FunctionalProperty>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="PUBLISHER.BOOK">
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="BOOK.ISBN">
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:domain rdf:resource="#BOOK"/>
</owl:DatatypeProperty>
```

Figure 5. A portion of resulting OWL document for the class BOOK and its properties.

In summery, this algorithm uses a key-based approach to generate ontology components. In general, concepts are created from tables, object properties are created from integrity constraints (foreign keys) and datatype properties are created from non-key columns. Concept hierarchy is built using a hypothesis based on primary and foreign keys (step 2). For semantically enriching the resulting ontology, some particularities of OWL language is exploited such as functional properties and inverses of object properties.

4.4 Mapping Generation

During the mapping process, a R2O [2] document is automatically generated to record the relationships between generated ontology components and the original database components. It includes (1) a full description of the database schema, (2) a set of concept map definitions consisting of the name of concepts with their

identifying column(s), and (3) a set of relation and attribute map definitions. This document can be used by the query web service to translate ontological queries into SQL queries and retrieve corresponding instances. Figure 6 shows an example of a class map definition from the mapping document.

```
<conceptmap-def>
  <name>BOOK</name>
  <use-table>BOOK</use-table>
  <described-by>
   <relationmap-def>
    <name>BOOK.PUBLISHERNO</name>
    <use-dbcol>BOOK.PUBLISHERNO</use-dbcol>
    <to-concept>PUBLISHER</to-concept>
   </relationmap-def>
   <attributemap-def>
    <name>BOOK.ISBN</name>
    <use-dbcol>BOOK.ISBN</use-dbcol>
   </attributemap-def>
  </described-by>
</conceptmap-def>
```

Figure 6. Mapping document for BOOK class and its properties.

5. DISCUSSION AND RELATED WORK

In the literature there are several approaches for addressing database to ontology mapping. As mentioned in section 2, these approaches can be classified into two main categories: (1) approaches that create new ontologies from existing databases and (2) those that map databases to existing ontologies. In the first category, we can note these relevant projects:

Volz et al. in [15][16] propose an approach based on semi-automatic generation of a F-Logic ontology from a relational database model. Mappings are defined between the database and the generated ontology. The ontology generation process takes in account different types of relationship between database tables and maps them to suitable relations in the ontology. The mapping process is not completely automatic and a user intervention is needed when several rules could be applied to choose the most suitable. The *DataGenie* [2] project is a Protégé [3] plug-in that allows the automatic generation of a Protégé ontology from a relational database. This generation process is simple and direct. Each table is transformed to a class and each attribute is transformed to a property. In addition, if the relational database table has foreign key references to other tables, these can be transformed to instance pointers, i.e. a new slot is added to the class representing the reference table whose value is an instance of the class representing the referenced table. The user selects manually the tables that he wants to map to the ontology, then the mapping process is done completely automatically. *Relational.OWL* [6] is an OWL ontology representing abstract schema components of relational databases. Based on this ontology, the schema of (virtually) any relational database can be described and in turn be used to represent the data stored in that specific database. This approach uses the meta-modelling capabilities of OWL-Full, which prevents the use of decidable inference on the resulting ontology.

In the category of mapping a database to an existing ontology, several languages have been proposed to formally express database to ontology mappings. *D2R map* [4] is a declarative, XML-based language to describe mappings between relational database models and ontologies implemented in RDFS. In D2R, basic concept mappings are defined using class maps that assign ontology concepts to database sets. The class map is also the container of a set of attribute and relation mapping elements called bridges. The D2R language allows flexible mappings of complex relational structures by employing SQL statements directly in the mapping rules. In [2], the authors propose another declarative language, called *R2O,* that describes mappings between database schemas and ontologies. It is more expressive than D2R map as it provides an extendable set of condition and transformation primitives. After the manual generation of a R2O document, it is processed by ODEMapster, a generic query engine that automatically populates the ontology with instances extracted from the database content. This operation can be done in two modes: (1) query driven, i.e. parsing a specific query and translating its result or (2) massive dump, i.e. creating a semantic RDF repository and translating the full database to it.

Beside languages, mapping approaches include some tools like *KAON Reverse* [4] which is a prototype for mapping relational database content to ontologies. The mapping rules describing the relation between the database schema and the ontology structure are defined manually, then the instances will be exported automatically. There are two principal types of mappings: Table Mapping relates a table to a concept while Column Mapping relates a table column to an attribute or to a relation. A column mapping can only be defined in the context of a Table Mapping. The whole mapping consists of a set of elements of these two mapping elements. The limitations of this tool are that it does not cope with multiple inheritance of concepts does not support relations with multiple domains, and does not support ontologies that have concepts with different namespaces. Another interesting tool is *Vis-A-Vis* [7] which is a Protégé plug-in that allows to map relational databases to existing Protégé ontologies. Mapping is done by selecting from the database a dataset corresponding to an ontology class. A new property is added to the class which consists of an SQL query which will be executed and return the desired dataset. This tool also performs a set of consistency checks to insure the validation of mappings.

Table 2 summarizes the features of these different approaches including our approach DB2OWL. We find that the definition of mapping is automatic or semi-automatic in the approaches that create a new ontology, whereas there is no approach allowing the completely-automatic definition of mapping to an already existing ontology. At the other hand, the process of ontology population is always automatic. We also note that the approaches that create a new ontology utilize the massive dump process for ontology population, except our approach DB2OWL which allows the query driven process.

Our approach belongs to the first category where a new ontology is created from the database, therefore we evaluate it versus the three first approaches. DB2OWL uses mapping rules similar to those of Volz et al. approach, but we use OWL instead of F-Logic, we consider the default cases of mapping in order to get a

[2] http://protege.cim3.net/cgi-bin/wiki.pl?DataGenie.

[3] http://protege.stanford.edu/.

[4] http://kaon.semanticweb.org/alphaworld/reverse/view.

full automatic process of mapping. We suppose that a user intervention may be needed later to refine the created ontology, but this still beyond the mapping process. In DataGenie and Relational.OWL the created ontology is a direct copy of the database schema and they do not take in count any specific table cases in the database. Furthermore, in Relational.OWL all database columns are mapped to datatype properties even the foreign keys, whereas in DB2OWL we map foreign keys as object properties.

Table 2. Features of different database-to-ontology mapping approaches.

Approach	Ontology		Exploitation		Automatisation	
	Created	Existing	Massive dump	Query driven	Mapping definition	Instance export
Volz et al.	×		×		Semi	Auto
DataGenie	×		×		Auto	Auto
Relational.OWL	×		×		Auto	Auto
KAON reverse		×		×	Semi	Auto
vis A vis		×		×	Manual	Auto
D2R map		×		×	Manual	Auto
R2O		×	×	×	Manual	Auto
DB2OWL	×			×	Auto	Auto

The major characteristic of DB2OWL is that it aimed at separating data mapping from schema mapping. Hence, the data manipulating, i.e. insert, delete, and update instances in the database, will not affect the corresponding ontology. We believe that a query driven population of the ontology is more effective than a massive dump, and it maintains the retrieved instances up-to-date.

6. CONCLUSION AND FUTURE WORK

We have presented an architecture for an ontology based cooperation system between heterogeneous information sources, and have focused on DB2OWL which is a tool to map relational databases to OWL ontologies. This tool is a local application encapsulated in the data provider service to create a local ontology from the local information source. We have implemented a prototype of this tool in Java that uses JDBC interface for database inter-connections. We use DatabaseMetaData java class to obtain a description of the database tables. These information about the database are encapsulated in a database model that we use as input to our mapping algorithm. The execution of this algorithm builds an abstract ontology model, which is implemented by the Jena API to give the OWL ontology (see figure 7). During the execution of the algorithm, a mapping document is automatically generated for recording the occurred correspondences between ontology components and their original database components.

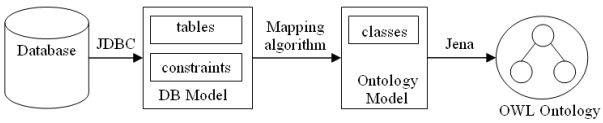

Figure 3. Mapping Process.

Currently, this tool deals only with Oracle and MySQL databases because they provide specific views about the database metadata (*USER_CONSTRAINTS* in Oracle, and *information_schema* in MySQL). Extension of the presented tool are underway to deal with other DBMS that provide such views. In addition, DB2OWL will be developed further to map several databases to one ontology.

7. REFERENCES

[1] Baader, F., Horrocks, I., Sattler, U. Description logics as ontology languages for the semantic web. In *Staab, S., Studer, R., eds.: Lecture Notes in Artificial Intelligence*. Springer Verlag, 2003.

[2] Barrasa, J., Corcho, O., Gómez-Pérez A. R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. *Second Workshop on Semantic Web and Databases (SWDB2004)*. Toronto, Canada. August 2004.

[3] Biron, P.V. and Malhotra A. (Eds). XML Schema Part 2: Datatypes. W3C Recommendation, May 2001. http://www.w3.org/TR/xmlschem-2/.

[4] Bizer, C. D2R MAP – A Database to RDF Mapping Language, The twelfth international World Wide Web Conference, WWW2003, Budapest, Hungary, 2003.

[5] Borgida, A., An, Y., and Mylopoulos J. Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences. In CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Cyprus, Part II, volume 3761 of LNCS, pages 1152 - 1169. Springer, 2005.

[6] de Laborda, C. P. and Conrad, S. Relational.OWL A Data and Schema Representation Format Based on OWL. In Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), volume 43 of CRPIT, pages 89 -96, Newcastle, Australia, 2005. ACS.

[7] Fuxman, A., Hernández, M.A., Ho, H., Miller, R, Papotti, P., Popa, L. Nested Mappings: Schema Mapping Reloaded. Proc. VLDB 2006 Conf., pp. 67-78, Seoul, Korea, 2006.

[8] Kalfoglou, Y., and Schorlemmer, M. Ontology mapping: the state of the art. Knowledge Engineering Review, 18(1), 1-31. 2003.

[9] Konstantinou, N., Spanos D., Chalas M., Solidakis E., and Mitrou N. VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents. International Workshop on Web Information Systems Modeling (WISM 2006) Luxembourg, 2006.

[10] Miller, R., Haas, L., and Hernandez, M.A. Schema Mapping as Query Discovery. Proc. VLDB 2000 Conf., pp. 77-88, Cairo, Egypt, 2000.

[11] Petrini, J. and Risch T. Processing Queries over RDF views of Wrapped Relational Databases. In 1st International Workshop on Wrapper Techniques for Legacy Systems, WRAP 2004, Delft, Holland, 2004.

[12] Prud'Hommeaux, E. and Seaborne, A,. SPARQL Query Language for RDF. World Wide Web Consortium, Working Draft WD-rdf-sparql-query-2006, 2006

[13] Rodriguez, J. B. and Gómez-Pérez, A. Upgrading relational legacy data to the semantic web. In Proceedings of the 15th International Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 1069-1070.

[14] Shvaiko, P., and Euzenat, J. A Survey of Schema-Based Matching Approaches. J. Data Semantics IV 3730 (2005), 146-171.

[15] Volz, R., Stojanovic L., Stojanovic N. Migrating data-intensive Web Sites into the Semantic Web. ACM Symposium on Applied Computing (SAC 2002). Madrid, Spain, March 2002.

[16] Volz, R., Handschuch S., Staab S., Studer R. OntoLiFT Demonstrator, 2004.

[17] Wache, H., et al. Ontology-Based Integration of Information - A Survey of Existing Approaches. In Stuckenschmidt, H., editor, IJCAI-2001 Workshop on Ontologies and Information Sharing, pages 108-117, Seattle, USA, April 4-5, 2001.