Fund Finder: A case study of database-to-ontology mapping

Jesús Barrasa, Oscar Corcho, Asunción Gómez-Pérez
(Ontology Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Spain
(jbarrasa@eui.upm.es, ocorcho@fi.upm.es, asun@fi.upm.es)

Abstract: The mapping between databases and ontologies is a basic problem when trying to "upgrade" deep web content to the semantic web. Our approach suggests the declarative definition of mappings as a way to achieve domain independency and reusability. A specific language (expressive enough to cover some real world mapping situations like lightly structured databases or not 1st normal form ones) is defined for this purpose. Along with this mapping description language, the ODEMapster processor is in charge of carrying out the effective instance data migration. We illustrate this by testing both the mappings definition and processor on a case study.

Keywords: database-to-ontology mapping, ontology population, information integration.

1 Introduction

It is a well known fact that there is a large quantity of existing data on the web stored using relational database technology. This information is often referred to as the Deep Web [Bergman, 2001] as opposed to the surface web comprising all static web pages. Deep Web pages don't exist until they are generated dynamically in response to a direct request. As a consequence traditional search engines cannot retrieve its content and the only manageable way of adding semantics to them is attacking directly its source: the database.

The case study presented in this paper has been developed in the context of the ESPERONTO[1] project. This project aims to bridge the gap between the actual World Wide Web and the Semantic Web by providing a service to "upgrade" existing content to Semantic Web content, retrievable and exploitable in an automatic and efficient way by Semantic Web tools. In this effort, ontologies play a key role, aiming at unifying, bridging and integrating multiple heterogeneous digital content.

The Fund Finder application is about migrating relational database content to the semantic web. Typically the input to this kind of problem is a database that contains the data to be migrated and an ontology that we want to populate with instances extracted from the database.

The important idea behind the approach described in this paper is that mappings between entities, relationships and attributes in the database's relational schema and the corresponding concepts, relations and attributes of the ontology will be defined declaratively in a mapping document. This mapping document will be the input of a processor charged of carrying out the effective migration in an automatic way. The fact of defining these mappings declaratively will make our solution domain independent and reusable.

The level of complexity of the mappings to be defined will depend on the level of similarity of the ontology's conceptual model and the E/R model underlying the database. Normally, one of them will be richer, more generic or specific, better structured, etc., than the other. This paper is organized as follows: Section 2 contains a description of the specific test case in which the study is based. Section 3 describes the system's architecture and components. Section 4 gives a global view of our approach to database-to-ontology declarative mapping definition and a set of possible mapping situations. Section 5 describes the most important features of the eD2R mapping description language. Section 6 describes how our work relates to other experiences and approaches. And finally, section 7 comments and evaluates the results and conclusions of our case study and gives a glimpse of some future trends.

2 Case study

The database we want to migrate (FISUB) contains incentives and funds provided by the Catalan and Spanish Governments and by the European Union, for companies or entrepreneurs located in the Spanish region of Catalonia. It contains more than 300 registers that are updated manually on a daily basis.

The reason why we want to migrate these contents to the Semantic Web is to be able to aggregate to them information from other web resources related to funding in the European Union and to allow web users to ask intelligent queries about funding resources according to some parameters like their profile, to look for complementary ones, to check compatibilities and incompatibilities between types of funding, and so on.

The FISUB database is very lightly structured as it stores almost all information on a main table called *FUND_OPP* (funding opportunity). This table has 19

[1] http://www.esperonto.net

columns and among them, the most important ones (which will be used for our examples) are the following:

- *TITLE* stores the name or accronym assigned to the funding opportunity.
- *BEGIN_END* stores important dates related to the funding opportunity as the beginning and end of validity.
- *LEG_REF* stores the legal announcement or approval of the funding opportunity.
- *FUND_OP_TYPE* stores a short description about the type of funding: A text in natural language describing whether it is a prize, a credit, a tax discount or other.
- *URL* stores the funding's home page if it has one.

Some other tables like *SECTOR* (activity sector) and *AIM* are used to add information about the activity sector covered and the objectives aimed by a funding opportunity. These satellite tables are linked to the main table *FUND_OPP* through standard foreign key fields. The main elements in the relational database schema can be seen in figure 1.

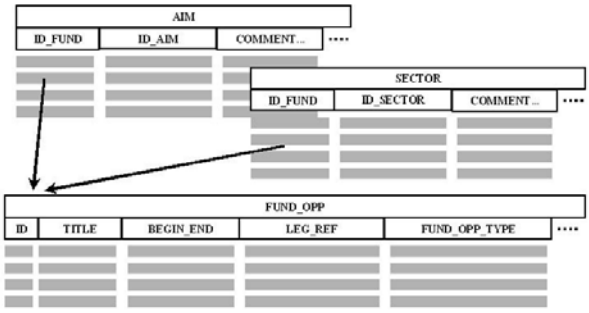

Figure 1: Excerpts from database tables.

The ontology to be populated is the Funding Opportunity ontology, which adds more structure and organization as well as enhanced inference and search capabilities to the legacy database. Figure 2 shows an excerpt of the ontology's concepts and relations.

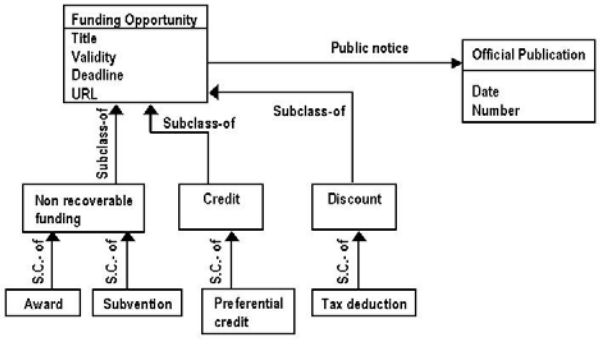

Figure 2: Excerpts from the Funding Opportunity ontology.

The mapping process is expected to extract instance data from the database and generate a set of instances committing to the funding opportunity ontology. Figure 2 shows graphically some of the expected results of this mapping. As can be seen, some record fields map directly their corresponding ontology attribute or relation (i.e. TITLE) but for some others this correspondence is not immediate (i.e. BEGIN_END) and some transformation is required. Let's have a look at some of these mapping situations.

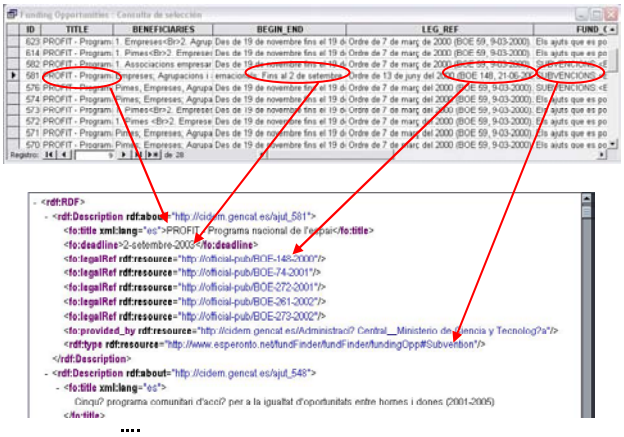

Figure 3: Results of the execution of the mapping between the FISUB database and the Funding Opportunity ontology.

- The TITLE field on the database maps directly the title property on the ontology because both refer to the same thing. The database field contains a string with the name or acronym that identifies the funding opportunity plus an optional short comment. In the example, "PROFIT" is the Spanish technical research support program.
- The BEGIN_END field on the database, needs to be transformed. It stores together the dates when the fund call opens and closes. In the ontology, the opening and closing dates are separate attributes, so some extraction needs to be done on the database field.
- The type of funding is determined by analysing the content of the field FUND_OPP_TYPE. If the keyword "subvention" appears in the field value, then the funding opportunity will be classified as a Subvention. If the keyword "prize" is found instead, then the type of the instance is Award, etc. As can be seen, keyword search particularly suits this case.
- The case of the LEG_REF data field is slightly more complicated. It stores a string referencing the (one or more) official publication in which the funding opportunity was proposed, approved, modified, cancelled, etc. by the competent authority. The corresponding element in the ontology is the LegalRef property and the fact of having more than one official publication mentioned into the LEG_REF field, which means the database is not in

first Normal Form (1NF), will lead to the generation of multiple relations to different instances from this single field value. As the official publications usually have alphanumeric codes as identifiers, regular expression evaluation seems adequate for this case.

3 System's architecture

Figure 4 resents the Fund Finder architecture. We have distinguished two layers: The modelling layer and the implementation layer (we are ignoring the formalism layer for the sake of clarity). At the first one we have the ontology conceptual model in the WebODE [Azpírez,2001] platform and the E/R model underlying the database. At the implementation layer, we have the ontology implemented in several ontology languages (OWL, DAML+OIL, RDF(S)…) using WebODE translators and the SQL implementation of the database relational model. An instance data sub-layer would contain instance data from the database (records) and instances of the ontology. The grey area in the figure shows the mapping definition and execution key elements.

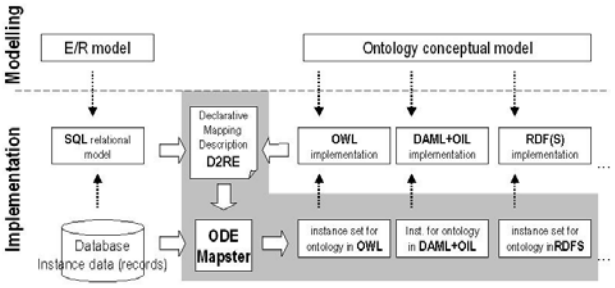

Figure 4: Diagram showing interactions between elements in our mapping approach.

- ***A declarative mapping description document: eD2R.*** This document contains the declarative definitions of the mappings between components in the SQL implementation of the relational database model and the ones in the ontology implementation. This documents is written in the eD2R mapping description language.
- ***The ODE Mapster processor*** is the software in charge of the mapping execution according to the directives of the aforementioned mapping document. The execution occurs automatically once the mappings are defined.
- ***A database*** containing the data to be migrated as instances of the ontology.
- ***An ontology*** to be populated with the data extracted from the database. The ontology can be expressed in any ontology implementation language, but instances of the ontology are generated in RDF in the first version of the processor.
- The automatically generated ***instance sets*** in RDF.

4 Global approach to database-to-ontology mapping

4.1 Declarative mappings

A declarative mapping is a set of explicit correspondences between components of two models. A mapping can be defined at different levels. In our case, it will be defined at the implementation level between a database's SQL description and an ontology's implementation. Furthermore, the intended direction of the mappings is from the database to the ontology, which means that we perform a process of data extraction from the database and we populate the ontology with the extracted information. That is why these correspondences will actually have the following form and not the other way round.

$OntologyComponent_i = Transformation(DatabaseComponent_j, DatabaseComponent_k ...)$

Where $OntologyComponent_i$ is any concept, attribute or relation in the target ontology and $DatabaseComponent_j$ is any database table or column.
A mapping between a database schema and an ontology can then be defined as a set of basic mapping expressions or *mapping elements* between components in both models like the one showed before. Inspired on the proposal of [Mena et al., 2001] and conveniently adapted to the specific case of databases, a basic mapping expression for a ***concept*** in the ontology will be defined as a 2-tuple <Rel, (a_1.. a_n)> where Rel is a SQL expression and a_1… a_n are columns of Rel that identify its objects (key columns). In other words, instances of concepts will be the records extracted from the database with an SQL query.

CONCEPT C1 : <Rel, (a_1..a_n)>

For an ***attribute*** or ***relation*** in the ontology, a basic mapping expression will be defined as a 4-tuple <Rel, (a_1.. a_n), (a_{n1}.. a_{nm}), f_{rl}> where Rel is a SQL expression; a_1.. a_n are attributes of Rel that identify its objects (the key columns); a_{n1}.. a_{nm} are columns of Rel that contain the attribute or relation values being mapped; and frl is a function f_{rl}:D_1x...xD_m → R that allows the transformation of the stored field data into the final values of the attribute or relation (D_i is the domain of field a_{ni} in the database and R is the range of the ontology's attribute or relation being described). In other words the value of an attribute or relation of the ontology will be obtained from one or more columns of an SQL expression directly or through the application of a transformation function.

ATT A1.1 : <Rel, (a_1..a_n), (a_{n1}…a_{nm}), fr_1>

The two mapping elements defined can be compacted in the following way:

CONCEPT C1 : <Rel, (a_1..a_n)>
ATT A1.1 : <(a_{n1}…a_{nm}), fr_1>
ATT A1.2 : <(a_{n1}…a_{nm}), fr_2> …

Where the ATT A1.i attribute mapping expressions inherit the two first elements (the SQL query Rel and the set of key columns a_1.. a_n) from their container CONCEPT C1.
What follows is an example of a mapping. We can see intuitively how the concept *FundingOpportunity* (the prefix *fo:* means that the concept is defined in the funding opportunity 'fo' ontology) maps all funding opportunities in the database marked as *new*. The mapping expression groups those records of the table *FUND_OPP* with value 1 in the field *NEW*. The different values of attribute ID identify the different records (ID is the key of the database table).
Within this concept mapping element a set of attribute or relation mapping elements can be defined. In the example the property *fo:title* maps directly the *TITLE* column and no function is applied to it's values. Attribute *fo:deadline* maps the *BEGIN_END* column after applying the function *getDeadline*. The same happens to the *fo:legalRef* relation, the column *LEG_REF* and the function *getLegalRef*. Functions used in the definitions should also be described in terms of the primitives provided by the mapping language being used, which will be discused later.

CONCEPT fo:FundingOpportunity :
*<[select * from FUND_OPP where*
FUND_OPP.NEW=1], FUND_OPP.ID>
ATTRIBUTE fo:title :
<FUND_OPP.TITLE, none>
ATTRIBUTE fo:deadline :
< FUND_OPP.BEGIN_END,getDeadline>
RELATION fo:legalRef :
< FUND_OPP.LEG_REF,getLegalRef>

4.2 Mapping cases

Based on the experience with the test case described in section 2, we have identified some mapping situations between the database implementation components and the concepts in the ontology. They are described and summarized in table 1. The second column in this table presents the database elements that can be mapped to an ontology concept, and the third column describes shortly the mapping case.

Table 1: Concept mapping cases

	Database implementation SQL element	Description
#1	View[2]	A view maps exactly one concept in the ontology.
#2	SELECT C1,…Cn FROM View	A subset of the columns in the view map a concept in the ontology.
#3	SELECT * FROM View WHERE f(C1,…Cn)	A subset (selection) of the records of a database view map a concept in the ontology.
#4	ImplicitSelect(View)	A subset of the records of a database view map a concept in the ontology but the selection cannot be made using SQL.
#5	T(Column)	One or more concepts can be extracted from a single data field.

Case #1 reflects the simplest mapping situation: The view in the database is semantically equivalent to the concept in the ontology and every record in the view corresponds to an instance of the ontology concept.

Case #2 is similar to case #1: the ontology concept and the database view refer to the same thing but two things may happen:

- The database view describes it with a higher level of detail by adding columns.
- In the view the relevant information for the specific concept we are interested in is merged with other concepts in the same view for optimisation purposes or just because of a bad structure of the database.

In **case #3**, the ontology concept is a subclass of the concept represented by the database table. The records in the database table being instances of the ontology concept can be extracted with an SQL query.

The same can be said for **case #4** with a peculiarity: the set of database records being instance of the ontology concept cannot be extracted with standard SQL and more complex techniques (i.e. keyword search, regular expression matching, natural language processing…) have to be applied on its data fields.

Finally **case #5** corresponds to situations in which a concept can be created out of a single column value. Or even more than one in the case of tables which are not in 1NF.

For ontology attributes and relations we have identified the following situations (the columns in table 2 are organized in the same way as those in table 1) :

[2] A view represents a single database table or any join of more than one table.

Table 2: Attributes and relations mapping cases

	Database element	Description
#1	Column	A column in a database view maps directly an attribute or a relation.
#2	T(Column)	A column in a database view maps an attribute or a relation after some transformation.
#3	n Column	A set of columns in a database view map an attribute or a relation.

Case #1 reflects the simplest mapping situation: both the column in the database is semantically equivalent to the attribute or relation in the ontology and share the same representation format. The correspondence is then direct.

Case #2 can cover three different cases:

1. The column in the database represents conceptually the same as the attribute or the relation in the ontology but they use a different representation format (i.e. currency unit transformation) and so the mapping needs a transformation function.
2. The column in the database stores the information needed to populate the ontology's attribute or relation but the information is mixed with other (noise) and it has to be extracted. Again a transformation function will be needed.
3. The same as the preceding one but furthermore, the column in the database stores more than one value (Not in 1NF) and each one of them needs to be extracted.

In **case #3**, the ontology's attribute or relation groups more than one database column. That means that the ontology property is less structured than its corresponding in the database. Let's take as an example the case of a postal address stored in a database using three columns one for the road name and number, another one for the postal code and a third one for the town name. These three fields would map one non-structured single field from the ontology containing the whole postal address resulting of the concatenation of the three column values in the database.

5 eD2R mapping description language

eD2R (extended D2R) is an extension of D2R MAP[3] which is a declarative, XML-based language to describe mappings between relational database models and ontologies implemented in RDFS developed at Freie Universität Berlin [Bizer, 2003].

D2R uses SQL statements in the mapping rules giving the possibility of handling highly normalized table structures, where instance data is spread over several tables. On the other hand, it fails to map low structured databases because of its limited expressiveness and we have enhanced with new primitives.

In D2R, basic concept mappings are defined using class maps. The class map is also the container of a set of attribute and property mapping elements called bridges (datatype property bridges and object property bridges respectively).

eD2R adds Operation and condition elements expressed in terms of elemental functions (Operation and Condition items) allowing the definition of complex and conditional transformations on field values based on techniques such as keyword search, regular expression matching, natural language processing and others. They cover all three case#2 attribute and relation mapping situations.

Classifier elements are used to apply what we called in section 4.2 Implicit Selections (selections which are not feasible via SQL queries) to classify elements in a taxonomy of concepts in the ontology.

Finally eD2R's field map elements are used for concept extraction from data fields and correspond to case #5 in the concept mapping cases table.

A detailed explanation of the eD2R mapping description language can be found at [Aguado, 2003]. The diagram in figure 5 shows the original elements in D2R and the ones in eD2R.

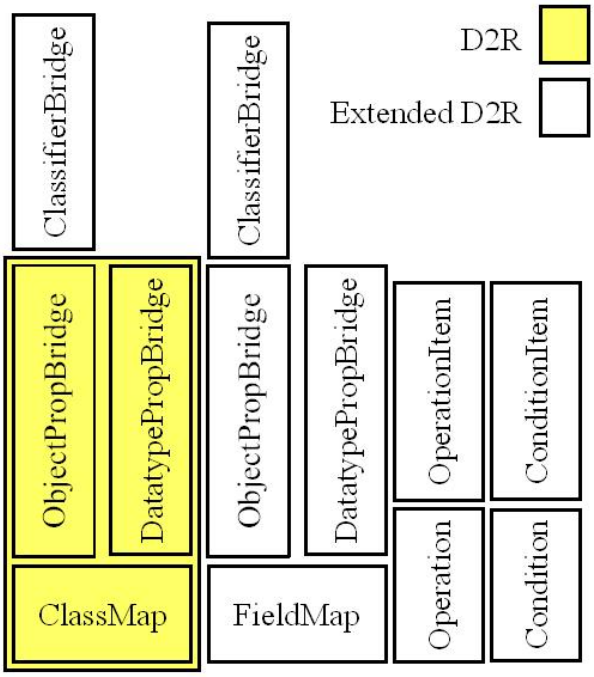

Figure 5: D2R and eD2R mapping description languages' elements.

[3] D2R MAP (Database to RDF) is available at: http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/D2Rmap.htm

6 Related work

Recent approaches like [Stojanovic et al.,2002] define mappings between a database and a ontology semi-automatically generated from the database's relational model. The level of similarity between both models is very high and mappings are consequently quite direct. They don't deal with complex mapping situations like the ones defined in section 4.2.

The same stands for REVERSE[4], an early prototype for mapping relational database content to ontologies, which is integrated in the Karlsruhe Ontology and Semantic Web Tool Suite (KAON).

[Handschuh et al., 2003] facilitates the manual definition of mappings, through the use of a server-side web page markup with information about the underlying database and its relation with the web page content (Web site cooperativity assumption). Their approach doesn't seem to deal with complex mapping situations like the ones tackled in this paper.

[Beckett and Grant, 2003] surveys and discusses mapping approaches to and from relational schemas.

Similar approaches to this work can be also found in the Intelligent Information Integration area, in which data from existing heterogeneous databases are extracted according to ontologies and then combined. Examples of such systems are Observer [Mena et al., 2000] and Picsel [Goasdoué et al., 2000], among others. The main differences with respect to our approach is that in these systems the mapping between the ontologies and the databases from which the ontology instances are extracted are not created declaratively but with ad-hoc software implementations.

7 Results, conclusions and future work

To sum up, the main outcomes of our experience are the following:

- The identification and characterization of a significant set of mapping situations when content stored in database is migrated into an ontology.
- Extension of D2R MAP with new features covering all the situations mentioned in section 2.
- Implementation of the ODEMapster processor to carry out the effective migration according to the definitions expressed using eD2R.
- Experimentation on a real world test case. The Fund Finder application.

Regarding the future trends of our work, intensive testing with other databases is being carried out and will continue as well as the enhancements to eD2R language.

The eD2R language has become quite complex as a counter-effect to its expressivity and the creation of a mapping document becomes a tedious, time consuming and error-prone task. A graphical user interface to support this activity is actually under development.

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

Acknowledgements

This work is partially supported by a FPU grant from the Spanish Ministry of Education (AP2002-3828), and by the IST project Esperonto (IST-2001-34373).
We would like to thank Raúl Blanco and Carles Gómara from CIDEM for providing the database and all information needed.

References

[Aguado, 2003] Aguado G, Barrasa J, Corcho O, Gómez-Pérez A, Suárez M, Blanco R, Gómara C. *Accompanying document to D8.3 Test Case application development. Fund Finder.* Esperonto project deliverable. October 2003.

[Azpírez,2001] Azpírez J, Corcho O, Fernández-López M, Gómez-Pérez A. *WebODE: a Workbench for Ontological Engineering.* First International Conference on Knowledge Capture (K-CAP01). Victoria B.C., Canada. October 2001

[Beckett and Grant, 2003] Beckett D, Grant J (2003) SWAD-Europe Deliverable 10.2: Mapping Semantic Web Data with RDBMSes. Technical report.

[Bergman, 2001] Bergman MK. *The Deep Web: Surfacing hidden value*. White paper. Sept 2001.

[Bizer, 2003] Bizer C. *D2R MAP – A Database to RDF Mapping Language*. 12th International World Wide Web Conference, Budapest. May 2003.

[Goasdoué et al., 2000] Goasdoué F, Lattes V, Rousset M (2000) *The Use of CARIN Language and Algorithms for Information Integration: The PICSEL Project*. International Journal of Cooperative Information Systems (IJCIS) 9(4):383–401

[Handschuh et al., 2003] Handschuh S, Staab S, Volz R. *On deep annotation.* 12th International World Wide Web Conference, Budapest. May 2003.

[Mena et al., 2000] Mena E, Illarramendi A, Kashyap V, Sheth AP (2000) *OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies*. International Journal on Distributed and Parallel Databases 8(2):223–271

[Mena et al., 2001] Mena E, Illaramendi A. *Ontology-based query processing for global information systems*. Kluwer Academic Publishers. Pags:86-88. 2001.

[Stojanovic et al.,2002] Stojanovic L, Stojanovic N, Volz R. *Migrating data-intensive web sites into the Semantic Web.* Proceedings of the ACM Symposium on Applied Computing, Madrid 2002.