IWONTO – SQL BASED MAPPING FROM RELATIONAL DATABASES TO ONTOLOGIES

Karsten Tolle and Mario Bachmann
Databases and Information Systems (DBIS), Johann Wolfgang Goethe-University of Frankfurt, Robert-Mayer-Strasse 10, D-60325 Frankfurt am Main, Germany

Keywords: Ontologies, Data Integration, Mapping.

Abstract: Ontologies are a formal representation of knowledge and semantics. With the emergence of the Semantic Web in the early 2000s, ontologies were pushed and from a theoretical point of view, many things could be solved. However, on the applicability level even finding the right tool to extract and load relational data into an existing ontology is a challenge and performing it remains a time consuming task. Based on various real case scenarios from the archaeological domain, we tried ontology-based approaches to integrate data. In those scenarios we had to deal with relational schema violating academic design principles. This complicated the situation. After an analysis of existing tools, we decided to implement our own tool to import relational data from different sources into our ontology. This paper will explain our motivation for implementing our tool called IwOnto and how it works. We decided to base it on SQL, which is known by our main target audience. Thereby we reached our design goal to avoid the necessity of learning another query language (e.g. SPARQL).

1 INTRODUCTION

Already since the very beginnings of databases back in 1970, the problem of integrating data from two or more databases has existed. The discrepancy can range from differences in the data model to the usage of different languages for describing the same content. Therefore, queries constructed for one database will not work for other databases, even if they contain the same logical content, e.g. archaeological data. Forcing people to use the same representation form, language and data models is not realistic.

One promising approach solving this is to move the problem from the technical to a higher abstraction level, e.g. by using ontologies. These create the vocabulary with which concepts and their relations can be defined and allow domain experts, as well as computer experts, to concentrate on the content and not on technical niceties.

We followed this approach and defined an ontology for our domain of archaeological findings, mainly coins. As next step we wanted to migrate existing data from different relational databases into this ontology in order to be able to export the new ontology enhanced with information from our relational data. The resulting ontology including the instances could then be saved as a file having a concrete syntax like OWL syntax to allow interchange with other applications.

We expected to find sufficient free tools that support this mapping and export process from relational data into an existing ontology. Surprisingly, we discovered that it turned out to be much more difficult to find a sufficient tool that is up to date and does not overwhelm the user. We finally decided to implement our own tool called *Integration with Ontologies* (IwOnto). One design goal was to base it on SQL, which is known by our target users and so avoids the need to learn another query language such as SPARQL.

In section 2 we will provide a state-of-the-art overview of existing tools we explored in the mapping area with their strengths and drawbacks. Details about IwOnto will be presented in section 3. This section includes some implementation background and a description of its features. Special emphasis is laid on how object properties are handled. Finally section 4 closes this paper with our conclusions and our planed future work.

Tolle K. and Bachmann M..
IWONTO – SQL BASED MAPPING FROM RELATIONAL DATABASES TO ONTOLOGIES.
DOI: 10.5220/0003934304300433
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 430-433
ISBN: 978-989-8565-08-2

2 EXISTING TOOLS FOR DATABASE-TO-ONTOLOGY MAPPING

There are powerful commercial tools like OntoStudio from Ontoprise, or SemanticWorks from Altova. However, we searched for a free, simple tool not overshadowed with too many features that could be used to map our relational data into our existing ontology. Due to design limitations of our relational data, an automatic approach was not an option. In fact, even for a human it was difficult to understand the underlying relational data without support of a domain expert.

During our search, we first came across the Protégé plug-ins DataGenie and DataMaster whereby DataMaster also supports Protégé-OWL. The main problem for both was that they support only the creation of a new ontology from given relational data. In the very latest version of DataGenie only pure bridge tables can be handled differently. This class of tools, which supports only the creation of a new ontology, does not fit our needs. Also the tool DB2OWL (Ghawi and Cullot, 2007) can be counted to this class of tools.

A very promising tool supporting the mapping to existing ontologies was VisAVis (Konstantinou, Spantos, Chalas, Solidakis and Mitrou, 2006). It is also a Protégé plug-in. The mapping and generation of classes and instances worked well. A less problematic issue was that VisAVis supports only the connection to MySQL and PostgreSQL databases. However, the main difficulty for us was the lack of support for object properties. The same seems to be true for RDB2Onto. It is claimed that the tool "*creates empty individuals in ontology from relational data*".

Some tools we tested where only partially executable, or even not executable at all. We invested a lot of time in getting the tool METAmorphoses (Svihla and Jelinek, 2005) running. METAmorphoses has a lot of library dependencies. Even by contacting the authors of the software we could not fix all problems. On top it has a promotion timer installed, which means that you need to wait two minutes before the tool starts.

We also had problems dealing with DERI Ontology Management Environment (DOME) and DIP Ontology Management Suite (OMS). A closer look at the web site of these tools reveals that there is now little activity involving them. In the DOME news section the last entry is dated to the middle of 2006.

Other research projects concentrated on defining a mapping language such as D2R (Bizer, 2003) and R_2O (Barrasa, Corcho and Gómez-Pérez, 2004). For R_2O exists the tool ODEMapster implementing it. ODEMapster is designed as a plug-in for the Neon toolkit which is based on Eclipse. The Neon toolkit was generated within the Neon Project – a European FP 6 project – that was finished in 2011. ODEMapster is a nice tool and provides an easy to use graphical way to map between the database and ontology side. When the mapping is straight forward this is a great tool, however, when the database schema – as in our case – is not conform to the ontology it gets very confusing. Additionally the mapping for object properties is not very well assisted by the GUI. When executing the mapping we often run into an error message reporting "*Unknown error occurred*" which did not helped at all.

D2R is implemented by the D2R Server. Additionally there is SquirrlRDF. Both of them are very powerful; however, they therefore do not transfer well to a simpler scenario as ours. The main purpose of these two is to enable SPARQL access to existing relational data. This was beyond our scope and we aimed to keep the user clear of SPARQL.

The following table provides a quick overview of the tools we explored together with a small comment.

Table 1: Overview of tools we explored.

Name	Comment
DataGenie	supersede by DataMaster
DataMaster	does not support existing ontologies
DB2OWL	does not support existing ontologies
VisAVis	no support of object properties
RDB2Onto	no support of object properties
METAmorphoses	promotion timer and difficulties with dependencies
DOME	outdated; problematic to run
OMS	outdated; problematic to run
ODEMapster	good for straight forward mappings; problems with object properties
D2R Server	SPARQL oriented; too heavy for our case
SquirrlRDF	SPARQL oriented; too heavy for our case

Under the mantle of the W3C we also found a working group trying to standardize the language for mapping relational data into RDF and OWL, called *RDB2RDF Working Group*. They generated a survey report – the latest version is from 2009 – providing a more general overview including approaches that

were out of our scope.

3 INTEGRATION WITH ONTOLOGIES (IWONTO)

The mapping tool *Integration with Ontologies* (IwOnto) was developed to provide a simple and useful method to perform information integration requiring only SQL skills (Bachmann, 2011). The development was based on the latest OWL API 3 for handling the ontology side. On the database side it relies on the JDBC API and is therefore flexible and not bound to a particular database system.

The ontology world distinguishes between two kinds of properties: data properties and object properties. A *data property* carries concrete data in the form of a value of a data type such as string, integer or float. When defining the mapping, the data properties can be mapped to one or more attributes of the intermediate table.

Object properties are different. The *object property range* is a class. They do *not* carry data, but they represent a reference to another object. This means they are the glue within the network of objects. This in fact is the most important issue about ontologies, because it allows us to view objects from a different viewpoint, and not just in tables.

In the database world the object properties correspond to the relations within the entity relationship model (ER-model). When translating it into the relational world of tables, we discovered that this translation is not unique. Two entities that have a relation can be translated into one, two or even three tables depending on the cardinalities. If everything goes well, we should have foreign keys in order to link the different tables. However, foreign keys are also used for situations where tables (objects) got split up into various tables in order to reduce redundancy – as done in normalization process. Additionally, the foreign keys are used to represent hierarchies in the relational world. Even for a person that can gain some semantics from the different table and attribute names, it is not easy to reconstruct the correct ER-model from a given relational schema.

In our situation it was even worse, because the existing relational database was not designed by database experts. It grew over time and was adopted on the fly to new requirements. Therefore, some referential integrity was not even defined by foreign keys and we also had the situation the representation of different objects was combined into one table. Therefore, an automatic reverse engineering by advanced applications would not help much.

Since other applications where based on the existing schema, a redesign of it was not an option. For this reason the mapping process of IwOnto is based on intermediate tables defined by an SQL statement. This way some of the problems - e.g. the mix of different entities within one table - could be handled using the right SQL query.

For the underlying mapping process of IwOnto the user passes through the classes of the ontology he wants to export. For each class he defines a separate SQL statement in order to build an intermediate table – e.g. in order to combine relational information that has been splitted up in order to gain normalization. This table is then the basis to define the mapping between data and object properties of a particular ontology-class and relevant attributes or attribute combinations on the database side. This also means one can perform pre-processing and cleanups, or include constraints using the power of SQL. On top the user can define functions for the mapping itself that can be used for conversions between different metrics, e.g. to convert from feet to meter or vice versa. As shown in Figure 1 below the compatibility of data types is checked by IwOnto in order to avoid invalid mappings.

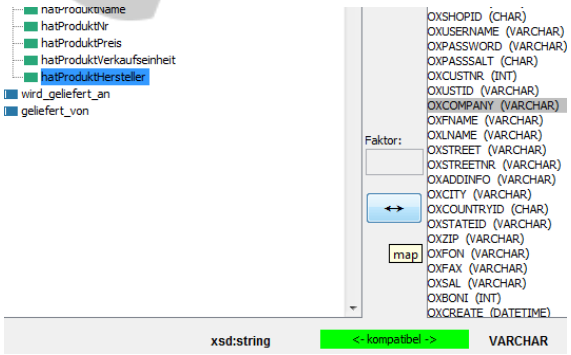

Figure 1: The GUI of IwOnto showing the ontology properties on the left and the database attributes of the intermediate SQL table on the right.

The mappings defined between the intermediate SQL table and the properties of the ontology are stored and listed in a separate window. Here the user has the ability to remove mappings selectively for the related class, or to remove the entire mapping for it performed to date.

When the mapping is executed for each row of the SQL table, an instance of the mapped class with the according attributes for the data properties is generated.

In order to link two objects with an object

property these objects must exist in order to be able to be referenced by object properties. This is not trivial which is reflected by the fact that various tools explored and described in the previous section do not provide this feature at all.

In order to solve this problem our mapping in IwOnto is split into *two phases*. In the *first phase* all needed objects for an object property should be generated. Because of possible dependencies this again is not trivial. If class A has an object property which range is class B, this implies a dependency of class A on class B. When building the dependency graph there can be chains and circles in it. To handle circles at least one link must be broken by removing one object property in the circle. This results in the loss of information that should be included in a later stage.

In the *second phase* the generated objects can be used. This is also not as easy as it might sound. One needs to identify the generated objects of the first phase clearly and without ambiguity based on the information provided by the intermediate table of the second phase. We currently have different approaches we implemented and tested to do so: a) based on the name of the object that needs to match the mapped attribute value, b) by a special data property as identifier for it, or c) a more fussy solution scanning all data properties in order to find a match. Currently we favour the first approach based on the name of the object because it is the most secure approach and this way we do not generate an IwOnto specific solution. However, none of these approaches is perfect, and we are also investigating additional and more flexible ways, e.g. based on constraints, to map from the attribute values of the intermediate table to objects already generated.

4 CONCLUSIONS AND FUTURE WORK

Existing mapping and export tools for migrating relational data to an existing ontology did not match our needs. We therefore implemented IwOnto. There are two main differences compared to existing tools a) it is based only on SQL and does not enforce users to adapt additional languages like SPARQL and b) the ability to move some of the mapping problems into intermediate SQL statements rather into the mapping itself, helps the user to split his problems into manageable portions.

It is already possible to handle object properties with IwOnto, which was problematic with many other existing tools. However, generating the according objects in case of longer chains is a time consuming task and does not fit our idea of an easy to use tool. Solving this in a sufficient and user friendly way is currently our main challenge.

The second main focus we have is the good usability of IwOnto. In the latest implementation we therefore improved the way how SQL statements can be entered. Now a statement can be tested and the user can see immediately the result of his statement in order to be sure he generated the correct intermediate table.

Currently we use IwOnto in the domain of archaeology and the size of relational databases we need to handle at the moment consists of just a few hundred findings. Therefore, performance issues are not our main focus. However, IwOnto is not bound to this domain and once we solved our current issues, we will have a closer look on performance. A first prototype of IwOnto is available under GPL 3.0 at SourceForge.

ACKNOWLEDGEMENTS

The authors would like to thank Steffen Försch, Marco Bender and Alexander Rezun for their contributions to IwOnto.

REFERENCES

Bachmann, M. (2011). Informationsintegration mit Ontologien am Beispiel von archäologischen Daten, Diploma Thesis at Goethe University Frankfurt am Main.

Barrasa, J., Corcho, Ó and, Gómez-Pérez, A. (2004). R2O, an extensible and semantically based database-to-ontology mapping language. In: Proceedings of the Second Workshop on Semantic Web and Databases, Springer-Verlag, Berlin, Germany, pp. 1069-1070. ISBN 978-3-540-24576-6.

Bizer, C. (2003). D2R MAP - A Database to RDF Mapping Language. The Twelfth International World Wide Web Conference, Budapest, Hungary.

Ghawi, R. and Cullot, N. (2007). Database-to-Ontology Mapping Generation for Semantic Interoperability, 3rd International Workshop on Database Interoperability (InterDB 2007), held in conjunction with VLDB 2007, Vienna, Austria.

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

Svihla, M. and Jelinek, I. (2005). The Database to RDF Mapping Module for an Easy Semantic Extending of Dynamic Web Sites. IADIS International Conference WWW/Internet, Lisabon, Portugal.