RDB2OWL: a Practical Approach for Transforming RDB Data into RDF/OWL

Guntars Būmans
Department of Computing,
University of Latvia
Raiņa bulvāris 19, Rīga, LV-1586, Latvia
(371)29488729
Guntars.Bumans@gmail.com

Kārlis Čerāns
Institute of Mathematics and Computer Science,
University of Latvia
Raiņa bulvāris 29, Rīga, LV-1459, Latvia
(371)67213716
Karlis.Cerans@mii.lu.lv

ABSTRACT

RDB2OWL is a simple approach of mapping relational databases into independently developed OWL ontologies. The approach is based on creating a mapping RDB schema, filling it with mapping information from which SQL scripts are generated that perform the instance-level transformation. We describe the RDB2OWL mapping schema and report on successful application of the technology to the migration of Latvian medical registries data.

Categories and Subject Descriptors

H. 4.4 [**Information Systems**]: Information Systems Application.

General Terms

Semantics.

Keywords

Relational databases, RDF triples, mappings.

1. INTRODUCTION

We describe an approach of generating RDF triples that correspond to a given target OWL ontology from a given source relational database (RDB). The approach is based on creating a mapping DB schema, filling it with appropriate mapping records, followed by generation of SQL sentences that generate the target RDF triples from the source RDB data. We report on the practical result of successfully transforming from RDB into RDF/OWL format the data from Latvian medical registries [1], [2].

A variety of approaches dealing with RDB to RDF/OWL data mapping already exist, some examples are R_2O [3], D2RQ [4], Virtuoso RDF Views [5], Triplify [6] and DartGrid[7]. The closest approach to ours is that of R_2O [1] where a similar principal schema of employing the SQL engine for implementing the declaratively specified mappings is used. The contribution of our demonstration is in presenting a SQL-based RDF triple generation framework where the SQLs sentences are generated, not manually written as in e.g. [6], the advanced generation options can be set, (e.g., "required properties"- generate RDF instance triples only if property instances exist for them); and a mapping validations can be performed by means of SQL. We believe that some our mapping solution patterns may be suitable for reuse in other RDB-to-RDF/OWL mapping formalisms.

2. RDB2OWL Framework

The RDB2OWL mapping establishes a correspondence between elements of a RDB schema and OWL ontology (or RDF schema) so that RDB record set could be migrated into RDF triple set. The mapping may include several database schemas and/or OWL ontologies. Fig. 1 illustrates the RDB2OWL framework architecture.

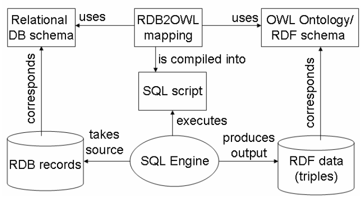

Figure 1. RDB2OWL framework architecture

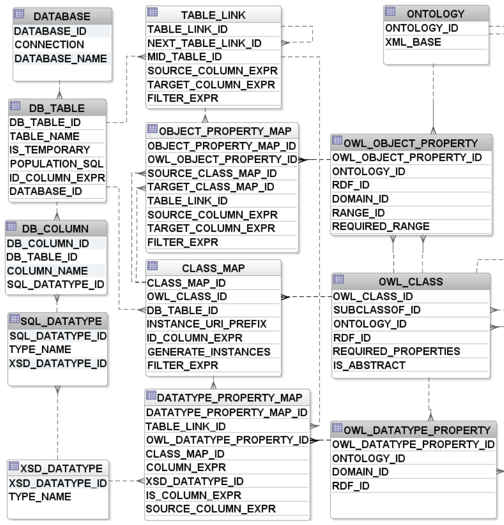

Figure 2. Mapping schema

We assume that the relational database and OWL ontology are given - this is a typical situation where OWL ontology is an end-user-oriented representation of the data contained in RDB. Figure 2 shows relational database schema that stores the RDB2OWL mappings. The mapping information relies on the source database schema description stored in tables *db_database, db_table,* and *db_column* and the target OWL ontology or RDF schema description stored in tables *ontology*, *owl_object_property* and *owl_datatype_property*. Only part of database or ontology information is stored in the mapping schema. The URI of ontology entities is obtained by concatenating *ontology.xml_base* field with *rdf_id* field from *owl_class, owl_datatype_property* or *owl_object_property* tables.

The mappings are specified in records of tables *class_map*, *object_property_map, datatype_property_map*. For *class_map* record *x* we call *base table* of *x* a source database table that is specified in *db_table* record linked to *x*. Each record *r* in the *class_map* table specifies triple generation of the form <*s,'rdf:type',o*>, where *o* is URI of the *owl_class* record linked to *r*. The URI of the subject *s* in the above triple is formed using the *instance_uri_prefix* in *r*, concatenated to the value of the expression specified in *r* in *id_column_expr* evaluated in records of the base table *t* of *r*. If *filter_expr* is specified in *r*, then only those records of *t* that satisfy it are considered for the subject's *s* generation. There are possibly several *class_map* records corresponding to a single *owl_class* record. The *class_map* records with *generate_instances=0* are not used for the triple generation but may later be referenced from property maps.

A record *ro* of *object_property_map* table specifies generation of triples <*s,p,o*> where the predicate *p* is the URI of the *owl_object_property* record linked to *ro*. We let *src* and *trg* to denote the *class_map* records that are referred in *ro* via the *source_class_map_id -> class_map_id* and *target_class_map_id -> class_map_id* links, respectively. Let, furthermore, *t_src* and *t_trg* be the base tables of *src* and *trg*, respectively. The *s* and *o* values in the above triple are obtained from all rows in the join of *t_src* and *t_trg* on the equality of columns specified in *ro* fields *source_column_expr* and *target_column_expr*, further filtered by *ro*'s *filter_expr*. Similarly, a *datatype_property_map* record *rd* specifies generation of triples <*s,p,o*>, where *p* is URI from the linked *owl_datatype_property* record. Let *src* be the *class_map* record that is linked to *rd*. Then *s* and *o* are obtained from each row of *src*'s base table – *s* by means of class map URI formation and *o* as a value of *ro*'s *column_expr* expression.

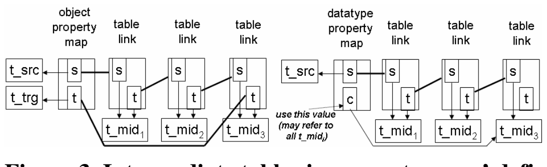

Figure 3. Intermediate tables in property maps' definitions

The *table_link* table allows introducing intermediate steps in table joining in object property definition, as well as auxiliary linked tables in datatype property definition. Figure 3 sketches the table linking schema in case of *table_link* usage (s stands for *source_column_expr, t* for *target_column_expr* and c for *column_ expr*; the arrow stands for column belonging to a table, and the bold line for equality condition; note that each iteration via *table_link* introduces a new table into the join expression).

3. A Simple Example

We consider a mini-university example, adopted from [8]. Figure 4 shows a RDB schema and OWL ontology of a miniature study registration system (in an UML/OWL profile [9] related notation). Observe the table splitting (*Course*, *Teacher*) and table merging (*Person* from *Student* and *Teacher*) in the ontology using the subclass relations, and OWL class *PersonID* based on a non-primary key column in each of *Student* and *Teacher* tables, and the n:n relation *takes* that reflects a student-to-course association that in the RDB is implemented using *Registration* table.

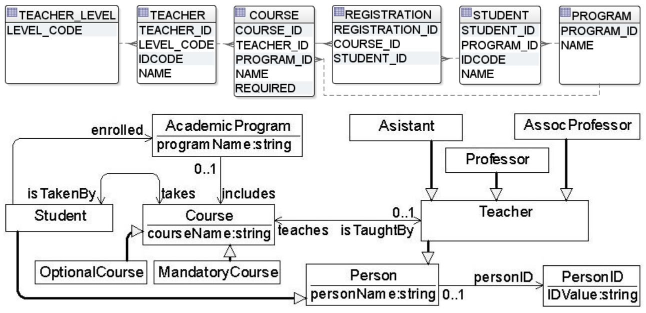

Figure 4. RDB schema and ontology of mini-university

The table below outlines some class map information (with linked OWL class *rdf_id* and DB table name; id denotes *class_map_id*).

id	**OWL class (rdf_id)**	**table_ name**	**filter_expr**	**id_col._ expr**	**instance_ uri_ prefix**	**gen._ inst.**
1	Teacher	teacher		teacher_id	Teacher	0
2	Student	student		student_id	Student	1
3	Course	course		course_id	Course	0
4	Mandatory Course	course	required=1	course_id	Course	1
5	PersonID	teacher		idcode	PersonID	1
6	PersonID	student		idcode	PersonID	1

The tables below outline some datatype and object property maps (*s* and *t* denote source and target class map id's).

s	**datatype_property**	**table_name**	**column_expr**	**filter_expr**
3	courseName	course	name	
1	personName	teacher	name	
2	personName	student	name	

s	**t**	**object_ property**	**table_ name(s)**	**table_ name(t)**	**source_ col_expr**	**target_ col_expr**
2	6	personID	student	student	student_id	student_id
1	5	personID	teacher	teacher	teacher_id	teacher_id
1	3	teaches	Teacher	course	teacher_id	teacher_id

Note that the maps for datatype and object properties can refer to class maps with id's 1 and 3 that are not used for class instance triple generation. The example has been further elaborated in [10].

4. Advanced Features

There are cases when a large database table, say *t*, is modeled by a set of OWL classes, each class *c* responsible for a certain subset of the table columns (e.g. there are different groups of measurements taken during a clinical anamnesis, all recorded into a single table). Mapping such table onto all the classes, would require writing lengthy filtering conditions involving all the group columns.

Our proposal is to introduce into the mapping schema explicit features allowing to specify generation of only those <*x*,'rdf:type',*o*> instance triples, where a generated triple <*x*,*p*,*y*> exists for some property *p* whose domain is *o*. This allows us to keep the simple mapping from *t* to all classes *c* associated to parts of *t* with no filtering. We specify this requirement by *owl_class.required_properties=1* and implement by deleting triples without property instances in the 2nd phase of the mapping generation. This feature is extensively used in the mapping definition for Latvian medical registries case (for 54 out of 172 OWL classes; 542 out of 814 OWL datatype properties on them).

The field *required_range* in *owl_object_property* table specifies requirement to delete those OWL object property instance triples <*x,op,y*> for which there are no <*y*,'rdf:type',*r*> triple with *r* being the range of *op*, after the *required_properties* optimization.

Auxiliary tables and temporary tables can be introduced allowing to use standard SQL for mapping specification. We have used a few auxiliary tables – the tables for classifiers not properly introduced in source database schema; and the Numbers table (with all numbers from 1 up to 999) used for field value splitting into multiple datatype property values (we used this pattern for 111 datatype property maps). Auxilary tables can be placed in a different database pointed to from *database* table row.

5. Implementation and Experience

The implementation of the mapping process in accordance to the mapping specification in mapping schema tables is obtained by "meta-level" SQL statements that generate SQL statements for RDB_data-to-RDF_triples transformation, one SQL statement per each row in tables *class_map*, *object_property_map* and *datatype_property_map*. The second phase of the mapping is implemented by SQL statements deleting the generated triples to meet the *required_properties* and *required_range* specifications.

The SQL statement generation by meta-level SQL statements works if intermediate *table_link* records in each mapping is bound by some k>0 (in our example *k=1* suffices); in the general case some procedural notation (e.g. T_SQL or jdbc) can be used.

We have applied the RDB2OWL approach to mapping of 6 Latvian medicine registries data (Cancer, Injury/Wound, Diabetes, Multiple Sclerosis, Narcotic patient, Psychic Behavioural Disorders) [1], [2]. The database size: 106 tables, 1353 columns and total 3 million rows, 3 GB of data. The OWL ontology had 178 non abstract OWL classes, 814 datatype properties and 218 object properties. The mapping database: 170 *class_map* rows, 832 *datatype_map* rows and 220 *object_property_map* rows.

The mapping has been implemented on a laptop computer with Intel Mobile Core 2 Duo T6570 processor running Windows Vista, 3 GB of RAM. The RDB2OWL database as well as source DB (Medicine DB) served by Microsoft SQL Server 2005. The triple generation process produced about 42.8 million triples in 18,5 minutes (6,5 for basic triple generation, 8 for indexing, 4 for *required_properties* filtering), what demonstrates satisfactory speed of SQL-based transformations for the practical use.

6. Mapping Validation

Since the mapping definition is stored in a RDB, validation is possible by using SQL. One can perform *Omission checks:* OWL classes or properties without corresponding class or property maps; DB tables not used in any class maps; DB columns not used in any column expression. *Consistency checks* may be used for property_map-to-class_map relation correspondence to property-to-class domain/range relation. The results of these checks are to be evaluated to decide, whether an error has been found, or the irregularity is by the mapping design. After loading the data into the target RDF data store, further checks of ontology data-to-schema consistency can be performed by means of SPARQL queries or invoking reasoners such as Pellet [11].

7. Conclusions

The outlined RDB2OWL mapping approach can be practically used in establishing a mapping between a source RDB schema and OWL ontology describing its data structure from the subject matter expert point of view, followed by the actual OWL ontology instance generation from the source RDB data. The approach has been tested on the practical example. It should be possible to translate most of RDB2OWL mapping constructions to other RDB-to-RDF/OWL mapping formalisms, including D2RQ [4] and Virtuoso RDF Views [5]. We note, however, that the translation of *required_properties* would substantially increase the size of the mapping and reduce its readability. Furthermore, we enjoy the easy extensibility and efficiency offered by RDB/SQL for mapping definition, implementation and validation.

8. REFERENCES

[1] Barzdins, G., Liepins, E., Veilande, M., Zviedris, M. (2008). Semantic Latvia Approach in the Medical Domain. In: Proc. of the 8th Intl Baltic Conference DBIS. Haav, H.M., Kalja, A. eds.). Tallinn University of Technology Press, pp. 89-102.

[2] Guntis Barzdins, Sergejs Rikacovs, Marta Veilande, and Martins Zviedris Ontological Re-engineering of Medical Databases, Proceedings of the Latvian Academy of Sciences. Section B, Vol. 63 (2009), No. 4/5 (663/664), pp. 20–30.

[3] Barrasa,J., Gómez-Pérez, A, Upgrading relational legacy data to the semantic web, In Proc.15th Intl. conference on World Wide Web Conference (WWW 2006), pages 1069-1070, Edinburgh, United Kingdom, 23-26 May 2006.

[4] D2RQ Platform www4.wiwiss.fu-berlin.de/bizer/D2RQ/spec

[5] C. Blakeley, "RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping)", OpenLink Software, 2007.

[6] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: Light-weight linked data publication from relational databases. In: Proceedings of the 18th International Conference on World Wide Web (WWW) 2009

[7] Discovering Simple Mappings Between Relational Database Schemas and Ontologies", Hu, W., Qu, Y., In Proc. of 6th Intl Semantic Web Conference (ISWC 2007), LNCS 4825, pp. 225-238, Busan, Korea, 11-15 November 2007.

[8] Barzdins.G, Barzdins.J, Cerans.K.: From Databases to Ontologies, Semantic Web Engineering in the Knowledge Society; J.Cardoso, M.Lytras (Eds.), IGI Global, 2008 (ISBN: 978-1-60566-112-4) pp. 242-266

[9] Ontology Definition Metamodel. OMG Adopted Specification. Doc. Number: ptc/2007-09-09, November 2007. URL: http://www.omg.org/docs/ptc/07-09-09.pdf

[10] G.Bumans, Mapping between Relational Databases and OWL Ontologies: an Example, to appear in Acta Universitatis Latviensis, (Scientific Papers University of Latvia), Computer Science and Information Technologies.

[11] Pellet, http://clarkparsia.com/pellet