An Automatic Database Generation and Ontology Mapping from OWL File

Jirapong Panawong[1], Taneth Ruangrajitpakorn[2,3] and Marut Buranarach[3]

[1]Faculty of Management Science, Nakhonratchasima Rajabhat University, Nakhonratchasima, Thailand
[2]Department of Computer Science, Faculty of Science and Technology, Thammasat University, Pathum Thani, Thailand
[3]Language and Semantic Technology Laboratory, National Electronics and Computer Technology Center, Pathum Thani, Thailand

Abstract. To promote ontology application development, some of the technical processes should be simplified with a supportive tool and an automatic method. This work presents a method to automatically generate a database schema from OWL file to prevent schema conflicts. Moreover, a mapping configuration can be created to associate an ontology and the generated data schema within the process. This method is designed to be compatible with an existing Ontology Application Management (OAM) Framework. With the proposed method, ontology labels are used to name data field name in database generation. The method allows any languages for ontology labels, but English ontology labels are recommended in this work since table names will be understandable. From testing, the produced mapping configuration to map ontology schema to database schema worked equivalently to human in terms of correctness but much faster in time consuming.

Keywords: Ontology application; Ontological database; Instantiation; Mapping Support

1 Introduction

In ontology application development, developers require to excel in several fields of expertise including knowledge engineering, semantic web technology, programing, inference engine tool, etc. In order to promote a development of ontology-based application, simplicity in processes is a must to help developers to reduce expertise requirements. In general, we can sum up processes as ontology creation, instance preparation, ontology-instance mapping, and developing application using those data and ontology. Each of these processes requires a good amount of time and effort to complete and verify.

Nowadays, several tools to support ontology application development are available. For ontology creation, an ontology editor such as protégé [1] and Hozo [2] are a great help, and their output as OWL format [3] file can accordingly be used in applications. Additionally, several inference engines to be used with ontology, such as Jena

[4] and JESS [5], are public and work smoothly. However, these tools still require practices in order to utilise them fully. In the recent date, a tool called Ontology Application Management (OAM) tool [6] was published and open for usage. The tool was designed to support in ontology application development in several processes for users who are an expert in a domain but are not excel in programing and technical concepts in semantic technology. These include 1) user-interface to map ontology and database schema as instantiation, 2) semantic search platform without background technical knowledge of related fields such as RDF [7], OWL and URI, and 3) recommender system platform with a simple user-interface to create rule without knowledge about inference engine syntax. The tool is a great help to a community of ontology application development as several applications in many domains were developed based on OAM tool, for example [8], [9] and [10].

Although OAM tool assists in many processes in development, one of the difficult parts for expert is the ontology-instance mapping. This part though is not a system-technical process; it is necessary and directly effect to overall performance of a semantic search and recommender application. Moreover, there are many cases that an ontology of the domain is created before data are gathered. Hence, this work aims to find a method to generate a database from OWL file and automatically map schema of the generated database to ontology schema. We expect that this will reduce burden of users to design a database compatible to their ontology and to map schema of both sources. To fill in instances, the design of database is exportable to a spreadsheet format file expected to be familiar and friendly to non-technical users. Lastly, this work is intentionally designed to fit compatibly with OAM tool.

2 Background

In this section, Ontology Application Management (OAM) tool is reviewed. The Ontology Application Management (OAM) framework is an application development platform aims to simplify creation and adoption of an ontology-based Semantic Web application [11]. This framework is an integrated platform that supports both RDF data publishing from databases based on domain ontology and processing of the published data in ontology-based Semantic Web applications, i.e. semantic search and recommender system applications. Moreover, the framework provides some reusable and configurable data and application templates customisable for different domain ontologies using configuration GUIs. The OAM framework introduces intermediate layers between user application and existing Semantic Web programming and development environment.

OAM framework uses ontology as a central structure for publishing RDF data from database and as a means to access the published RDF data. The layers introduced by OAM aim to hide complexity of the underlying Semantic Web data standards and models. Fig. 1 shows a layered architecture of the OAM framework.

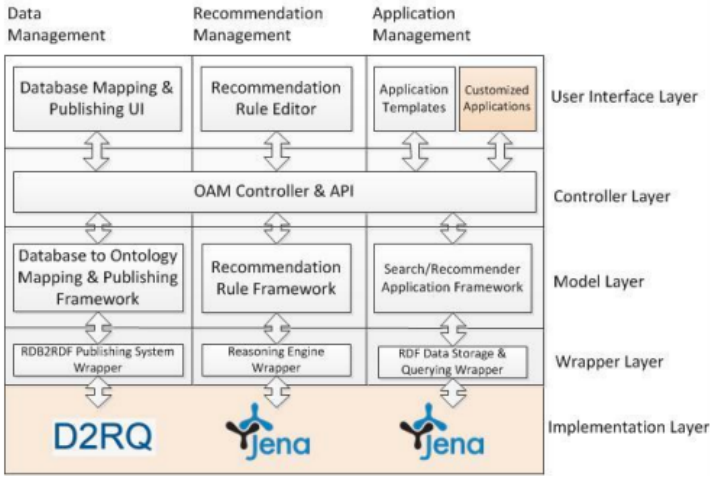

Fig. 1. A layered architecture of the OAM framework [11]

The framework is implemented on top of existing Semantic Web data and application platforms that are D2RQ [12], Jena's RDF data storage and Jena's reasoning engine [4]. The OAM framework uses relational database to ontology data mapping, recommendation and application templates on top of these systems. The user can use the provided management tools via web browsers (google chrome recommended) in creating and managing an ontology-based Semantic Web application. For advance usage, users of OAM can choose to use the Java API function in application development.

Moreover, a recent upgrade of OAM included a rule management using spreadsheet [13]. This allows the users to use their own vocabulary and spreadsheet application in managing recommendation rules. The framework was validated to be successful tools for supporting by being utilised in several ontology application projects.

3 Methodology

This work aims to create a database compatible with a given ontology file and automatically generate a configuration file of mapping for OAM tool. For an overview, there are three main processes shown in Fig. 2.

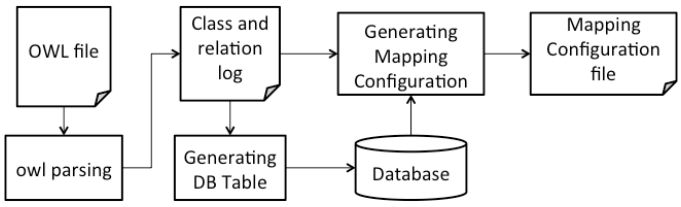

Fig. 2. An overview of processes for an automatic database generation and ontology mapping from OWL file

3.1 Reading OWL File

Firstly, we read the inputted OWL file [3] and parse the annotations of OWL. From OWL, we capture two kinds of ontological unit: concept and concept relation. For step-wise, we apply an algorithm shown in Fig. 3.

Algorithm: Parsing of OWL class and property

```
1   while unread class ≠ 0
2       read and record concepts URI
3       if class contains property
4           if property = data_property
5               record property name and its data_type
6               if cardinality of property >1
7                   mark the property for multi-value
8           if property = object_property
9               record property name and its class_constraints
10               if cardinality of property >1
11                  mark the property for multi-value
```

Fig. 3. An algorithm to parse OWL class and property

In detail of the algorithm, parsing starts from the root class (thing class for protégé or Any class for Hozo). Next, properties of the class are read and recorded. For properties, types of property are all the matter in further processes so object-property (part-of) and data-property (attribute-of) are recorded separately along with their constraints. Once we run through all properties, we search for sibling classes and subclasses and keep the iteration until all classes are read.

3.2 Table Generation

Since we aim to be compatible with OAM, MySQL relational database is chosen as our target format. With the captured classes and their relations, their existence is considered. For each class, a table is created, and their properties are generated as table columns. A data property is considered as a field to input data in while an object property is required to be linked to another table using a foreign key following a class constraint

For a multi-value property, an extra table is generated for either data or object property. Foreign key will be assigned instead of direct value. This multi-value constraint forms a many-to-many relation of data. A table name and column tags are generated following a label of an ontology class and relations. The generated database table and columns along with a label of the ontology class are written into a log file to be used in a further process. Please be noted that only own properties of a class are considered since inherited properties from the superclass can be handled by ontology processing in OAM. Moreover, the naming of tables and table columns can only be done with an otology with English label. For those ontologies without English label,

number will be generated instead for identification, such as *table1* and *column4*. An illustration of table generation is shown in Fig. 4.

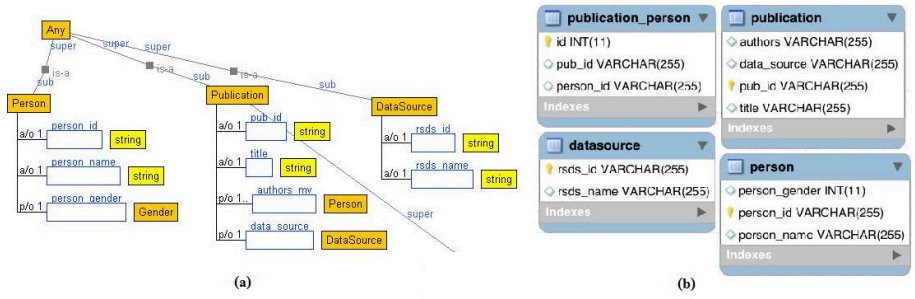

Fig. 4. Images showing ontology classes (a) and the generated database schema associated with the ontology (b)

From examples in Fig. 4-a, we can see that the ontology (from Hozo ontology editor) contains a main class "*Publication*" with four properties. Let us focus on this class first. For two data properties (indicated as A/o) of the class, two columns of table '**publication**' shown in Fig. 4-b are created for storing the string value for "*pub_id*" and "*title*" relation. These database columns are generated with a data type '**VARCHAR**' according to ontology data type '*string*'. An object property relation (marked as P/o) to relation "*data_source*" is generated to another column field required for a foreign key to link to another table '**datasource**' generated in association with "*Data_Source*" class from the ontology. Last, a class "*Person*" in ontology is generated into another separated table '**person**' in database shown in Fig. 4-b. This relation, however, is marked with multi-value (as '_mv' after relation name) since its cardinality is 1 or greater (shown as '1..' in ontology editor interface in Fig. 4-a). This requires many-to-many relation; thus, another table is specially created as '**publication-person**'. In this separated table, id of **publication** instance and id of **person** instance are linked.

3.3 Generating a Mapping Configuration

From the log file kept information about original ontology class and its generated table, a mapping configuration can be generated. Since a database schema is directly created from ontology schema, we would not find a case of a conflict of schemas. For a template of a mapping syntax, we follow OAM mapping configuration. By adding ontology label and the name of table/table column, we gain an automatic way to generate a mapping configuration compatible with OAM. Fig. 5 shows a generated mapping configuration file.

```
insert into aclass ( cname, unique_property, class_type)
  values( 'DataSource', 'rsds_id', 0);

insert into aclass ( cname, unique_property, class_type)
  values( 'Gender', 'gender_id', 0);

insert into aclass ( cname, unique_property, class_type)
  values( 'Person', 'person_id', 0);

insert into aclass ( cname, unique_property, class_type)
  values( 'Publication', 'pub_id', 0);


insert into mapping ##( mid, table_name, class_name, unique_property, primary_key, pkey_la
  values(1, 'datasource', 'DataSource', 'has_rsds_id', 'rsds_id', 'DataSource', 0);

insert into mapping ##( mid, table_name, class_name, unique_property, primary_key, pkey_la
  values(2, 'gender', 'Gender', 'has_gender_id', 'gender_id', 'Gender', 1);

insert into mapping ##( mid, table_name, class_name, unique_property, primary_key, pkey_la
  values(3, 'person', 'Person', 'has_person_id', 'person_id', 'Person', 0);

insert into mapping ##( mid, table_name, class_name, unique_property, primary_key, pkey_la
  values(4, 'publication', 'Publication', 'has_pub_id', 'pub_id', 'Publication', 0);
```

Fig. 5. Some parts of a generated mapping configuration file following examples from Fig.4

However, this method can only generate a mapping configuration following original OWL and the generated table schema. For editing, users are asked to make change of the mapping via OAM user interface.

4 Experiments and Discussion

To approve a potential of the proposed method, we set up two experiments. The first one is to check an accuracy of an automatically generated mapping configuration by comparing to a mapping done by human. The second experiment is to examine how supportive this method give based on time consumption in mapping ontology-data.

For the first experiment, we asked a user who has experience with OAM to perform mapping of ontology to the generated database. The given ontology contains 61 classes, 18 object properties and 27 data properties. We then compare a mapping configuration file from human and our generated mapping configuration file. In comparison, class mapping and relation mapping are separately counted. The comparison result is shown in Table I.

Table I. A comparison result of mapping between human and automatic generation

Type	**Accuracy**	
	Amount	**Percentage**
Class mapping	61/61	100%
Object property mapping	17/18	94.44%
Data property mapping	27/27	100%
Sum	**95/96**	**98.96%**

From the result shown in Table I, performance of the proposed method was reliably good. The only difference of the mapping result was the case of a cardinality of property is one or more. For the case, human mapped this class constraint to the target class and all of the subclasses while the generated mapping only gave a mapping to the target class. However, the outputs of both would give the same result in applications since ontology processing in OAM can handle the tree spanning based on the given OWL. Thus, the result though is different, but the both mappings are legit and acceptable.

The second experiment was set to compare time used for mapping. Three ontologies are chosen to represent an effect based on ontology size. Details of the three ontologies are given in Table II.

Table II. Details of three ontologies as a size representative for time-wise experiment

Ontology Size	Classes	Object Properties	Data Properties
Small	15	5	5
Medium	61	18	27
Big	183	82	71

In this experiment, time is counted in minutes used in only mapping process while a fraction of minutes is considered as a minute. The mapping includes class mapping, subclass mapping and property mapping. A person to map these ontologies is a user who has used OAM in his 2 projects and continuously been using OAM for over 11 months. A result of time consumption in mapping between human and the proposed method are given in Table III.

Table III. A result of time consumption in mapping

Ontology Size	Human	Proposed Method	Time difference
Small	38 minutes	1 minutes	36 minutes
Medium	109 minutes	2 minutes	107 minutes
Big	192 minutes	5 minutes	187 minutes

From the results in Table III, we can see that time in use for mapping by human was much higher than the automatic method. However, we notice that the time consuming in average per class was lower the more he mapped since the most time consuming period was when he started mapping by looking through and learning of ontology classes and database. On the other hand, the automatic method skipped the understanding process and went directly to labels.

For further validation, those generated mapping configuration files with some data in the database were tested in OAM semantic search application, and they worked fine in actual usage. This proves that the generated mapping can help in a development of an ontology application. For a summary from the experiments, the proposed method can perform equivalently to human in terms of quality but much greater in time consuming.

5 Conclusion

In this work, we propose a method to support ontology application development by automatically generated database schema and the ontology-data mapping. It was designed to be compatible with OAM framework to support users who may not excel in programing and technical implementation. The method requires an OWL file from ontology editor to generate a database schema along with a mapping configuration file. The method is though applicable to any language of ontological label, but English language is recommended for easier modification in further usage. From the testing, the method can yield a high quality mapping of ontology-data schema with much less time consuming comparing to human.

References

[1] protégé: available online at http://protege.stanford.edu/
[2] Hozo Ontology Editor: available online at http://www.hozo.jp
[3] McGuinness, D., Harmelen, F., OWL Web Ontology Language Overview: available online at https://www.w3.org/TR/2004/REC-owl-features-20040210/
[4] Apache jena: available online at https://jena.apache.org
[5] JESS, the Rule Engine for the Java™ Platform: available online at www.jessrules.com/jess/download.shtml
[6] Buranarach, M., Ruangrajitpakorn, T., Anutariya, C., Wuwongse, V.: Ontology Design Approaches for Development of an Excise Duty Recommender System. In: Kawtrakul, A., Laurent, D., Spyratos, N., and Tanaka, Y. (eds.) Information Search, Integration, and Personalization. pp. 119–127. Springer International Publishing (2014).
[7] Bruijn, J., Welty, C., RIF RDF and OWL Compatibility: available online at https://www.w3.org/TR/2013/REC-rif-rdf-owl-20130205/
[8] Wongpatikaseree, K., Ikeda, M., Buranarach, M.: Activity Recognition using Context-Aware. Infrastructure Ontology in Smart Home Domain. In The Seventh International Conference on Knowledge, Information and Creativity Support Systems (KICSS2012) (2012).
[9] Chariyamakarn W., Boonbrahm P., Ruangrajitpakorn T., Supnithi T. : An Ontology-based Supporting System for Integrated Farming towards a Concept of the Sufficiency Economy. The International Joint Conference on Computer Science and Software Engineering (JCSSE2016) (2016).
[10] Somsuphaprungyos S., Boonbrahm S., Ruangrajitpakorn T. : An Ontology-based Framework of Intelligent Services for Smart Campus. In The Tenth International Conference on Knowledge, Information and Creativity Support Systems (KICSS2015) (2015).
[11] Buranarach, M., Thein, Y., Supnithi, T.: A Community-Driven Approach to Development of an Ontology-Based Application Management Framework. In: Takeda, H., Qu, Y., Mizoguchi, R., and Kitamura, Y. (eds.) Semantic Technology. pp. 306–312. Springer Berlin Heidelberg (2013).
[12] Bizer, C., Seaborne, A.: D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs. In: Poster at the the 3rd International Semantic Web Conference (ISWC2004) (2004).
[13] Buranarach, M., Rattanasawad, T., Ruangrajitpakorn, T.: Ontology-based Framework to Support Recommendation Rule Management using Spreadsheet. In The Tenth International Conference on Knowledge, Information and Creativity Support Systems (KICSS2015) (2015).