DE GRUYTER OPEN
1862 RIGA TECHNICAL UNIVERSITY
Information Technology and Management Science
ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2017, vol. 20, pp. 69–73
doi: 10.1515/itms-2017-0012
https://www.degruyter.com/view/j/itms

Database Concepts in a Domain Ontology

Henrihs Gorskis[1], Ludmila Aleksejeva[2], Inese Poļaka[3]
[1–3]*Riga Technical University, Latvia*

***Abstract* – There are multiple approaches for mapping from a domain ontology to a database in the task of ontology-based data access. For that purpose, external mapping documents are most commonly used. These documents describe how the data necessary for the description of ontology individuals and other values, are to be obtained from the database. The present paper investigates the use of special database concepts. These concepts are not separated from the domain ontology; they are mixed with domain concepts to form a combined application ontology. By creating natural relationships between database concepts and domain concepts, mapping can be implemented more easily and with a specific purpose. The paper also investigates how the use of such database concepts in addition to domain concepts impacts ontology building and data retrieval.**

***Keywords* – Database, data mapping, ontology.**

I. Introduction

Ontologies are a great tool for describing concepts and their meaning relative to other concepts. One definition of ontologies was given by Tom Gruber in 1993. By his definition, an ontology is a “shared formal specification of a domain conceptualization” [1]. Ontologies are used to define terminology for communication, data meta-modelling and other purposes. This is achieved by defining relations and meaning in addition to the hierarchy of concepts used within a certain domain. By sharing this terminology and the implicit meaning behind the terminology, ontology is knowledge sharing and communication. The model of concepts provided by the ontology, which is a meta-model of the underlying data, allows extending information by adding conceptual knowledge to the data provided in communication between two parties.

In the task of ontology-based data access, these fundamental capabilities of ontology descriptions are used for the extraction of data from databases. The goal of ontology-based data access is to be able to use ontology as a communication layer to the content of a database, to be able to use the capabilities provided by the ontology on existing data [2]. By accessing data through ontologies, it is possible to use the knowledge contained in the ontology on the data from the database, thereby extending it, as well as filtering it to conform to the concepts from the ontology. However, before ontology-based data access has become possible, the task of mapping ontology concepts to the data should be accomplished. Since the ontology description is very different from database schemas, some intermediate mapping is required. The present paper proposes a novel data mapping approach that uses the ontology concepts as mapping points to basic database objects. By using the ontology own elements as mapping points to the database, better ideological consistency of the knowledge in the ontology is also observed. This is one more benefit, in addition to simplicity, this mapping approach provides.

II. Existing Approach

Existing mapping solutions rely heavily on external mapping documents (Fig. 1). These documents contain mappings as rules and descriptions in additional files [3], [4]. Whichever software tool is used in the task of obtaining data from the database, it uses these additional files to know how to create data queries. This leads to unclarity of which ontology concepts are or are not connected to the database. The ontology engineer cannot immediately tell this information without having to consult a separate mapping document first. Using mapping documents is also conducive to an ontology-first approach when building an OBDA system. Using a mapping document is most comfortable when an ontology already exists. In this case, the mappings are created onto the existing concept. It is less comfortable when the database has to have an impact on a newly built ontology for a specific task. In this situation, the ontology and the mapping file must be created simultaneously in parallel.

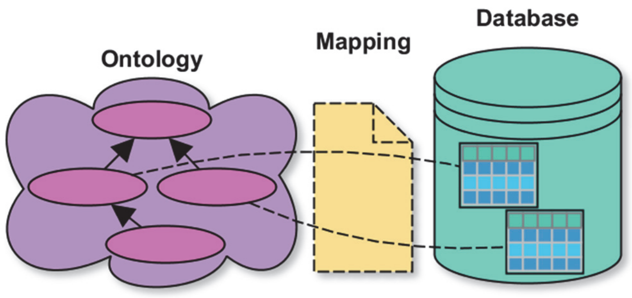

Fig. 1. Ontology to database mapping.

One example of the usage of mapping files is the software solution called Ontop [5]. Ontop uses a mapping file that describes how to obtain individuals from the database. The mapping file defines a target individual with a placeholder name, the individual’s class, the properties of the individual with placeholder values and SQL query. The placeholder name and any values are extended with the values obtained from the database. This way, Ontop creates a virtual RDF graph and combines it with the ontology. An example of a mapping for the IMBD movie ontology can be seen below [6].

```
mappingId  Actor
target   imdb:name/{person_id} a dbpedia:Actor .
source   select person_id from cast_info where
cast_info.role_id = 1
```

```
mappingId  Person has Birth Name
target   imdb:name/{person_id} dbpedia:birthName
{name} .
source   select name.id as person_id, name.name as
name from name

mappingId  Person has Birth Date
target   imdb:name/{person_id} dbpedia:birthDate
{dob}^^xsd:string .
source   select person_id, info as dob from
person_info where info_type_id = 21
```

Sometimes the mapping information is not provided in an additional mapping file, but using the other storage and usage solutions. One example is the usage of databases themselves [7] for the storage of the mapping rules.

Another approach to database mapping is to recreate the database structure as an ontology [8], [9]. In this approach, the database is taken as the basis of the ontology description (Fig. 2). The created ontology mimics the database. This has the advantage that data will fit well onto the ontology concepts. The downside to this approach is that the created ontology is mostly a database structure and provides little domain definitions. This is sometimes done not specifically for the task of ontology-based data access, but also for merging multiple databases [10], [11] or for other data migration and data validation [12] tasks.

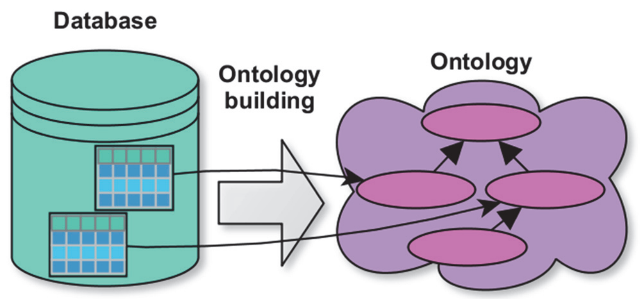

Fig. 2. Ontology creation from database structures.

One related approach that has some similarity to the approach proposed in the present paper is the use of concept annotations as indicators and instructions on how to obtain data from a database [13], [14]. In this approach, the mapping is performed by asserting annotations on ontology concepts. The annotations contain database queries as a plain text and they can be executed by a data retrieval engine. The data obtained by the query are individuals and values. This approach allows storing both domain and mapping information in the same location, namely the ontology. The approach described in the paper differs from all the related approaches.

III. Database Concepts

The present paper proposes the usage of special database concepts alongside natural domain concepts within a combined application ontology. These database concepts themselves are to be used as mapping points to database objects. This can be done using only the concept names provided as IRI strings. These IRI strings are able to contain sufficient information to point to database objects. Mapping can be implemented by correctly interpreting these IRI strings.

The proposed method provides three types of mapping concepts – table or view concepts, column concepts, and table relations. Table or view concepts are mapped by using classes. A table or view class points to a database table or view. Database records found in these tables can be interpreted as individuals of the respective class. Any database table record can be obtained as an individual of the table class. Database columns are mapped by using data properties. If the record itself (the fact of existence of the record) is an individual, then the columns of the table are data properties with specific values. An individual uses these special database data properties to indicate that the values were obtained from the database. Relations between tables are mapped by using object properties. The mapping of tables to class concepts, or table columns to data properties is not a new idea [15], [16]; however, the use of just the concept names themselves without additional mapping rules is novel.

These special database concepts are used to create a link to a database to obtain individuals and values. To be able to implement this, they must be distinguishable from the other non-database linked domain concept. This is done with the use of prefixes. A mapped ontology will have multiple different prefixes. All domain concepts use one or more domain prefixes. Additionally, a database prefix is created, for every separate database used in the mappings. For example, let us say that the ontology describes one domain and all the domain concepts are created within this ontology. Therefore, all domain concepts shall begin with the prefix "http://example.com/ontology/". This ontology is also mapped to and references one database. Therefore, all database concepts in the domain shall begin with the prefix "http://example.com/database/". The prefixes may also indicate the type of a concept. This can be done using the following example prefixes:

```
"http://example.com/OBDA/database1/table/",
"http://example.com/OBDA/database2/table/",
"http://example.com/OBDA/database1/view/",
"http://example.com/OBDA/database1/column/",
"http://example.com/OBDA/database1/relation/".
```

Apart from indicating the query creation system, the type of database object, the use of prefixes has another advantage. By using prefixes, it is possible to have multiple concepts with the same name. The ontology may contain a domain concept, a table concept, and a column concept, all with the same name, but different meanings. For example, when multiple databases are mapped to the ontology, they may contain tables with the same name. Providing the database name in the prefix of the IRI allows depreciating between multiple data sources. Another example is the case of a database table "person" that may contain a multitude of records related to people, including some test record, change tracking records or other records, which are stored in the same table, but do not describe real persons. There may also be a column in one or more tables with the name of a person. These columns may indicate a relation to the table person. Finally, the concept of person may also exist in the domain. The domain concept description of what a "person" is may differ from the data in the table person. Therefore, it is important to distinguish these concepts. The use of different prefixes makes this possible. This allows database concepts to be freely inserted into an ontology at any point in their respective element hierarchy, without disturbing the domain concepts.

IV. Ontology Building with Mixed Concepts

The availability of the proposed database concepts has also an effect on the process of ontology building. The database concepts can be used as a basis for the whole domain ontology. The available data from the database can shape how the ontology should be built. The ontology building process is modified to first inspect and analyse the available data and the structure of the database. Next, important tables, views, columns and values are identified. Based on these insights from the database, database-concepts are created in the ontology. After important database concepts are found and added to the ontology, the ontology can be extended by adding domain concepts above and below the database concepts in their hierarchy. Since all concepts from the database or another data source are distinguishable from domain concepts, the ontology engineer is free to use any domain terminology in the process of extending the knowledge contained in the ontology. Seeing the database concept in the same place as domain concepts helps create meaningful concepts and relation to the data. The creation of domain concepts that are not related to database concepts can indicate that these domain concepts may not be needed.

It is important to understand that the database concepts are not to be interpreted and used in the same way as domain concepts are. For example, a table concept, which is added to the ontology in the form of a class, represents the set of all known records in the table. No more and no less. It does not matter what the table is called or what use it is supposed to fulfil. The table "persons" is still just the set of records from the database providing information about people; it is not the representation of the concepts describing what a person is. If the actual concept of a person is needed, it must be added separately to the ontology as a domain concept. However, the concept of a person will be related, in some way, to the database table concept. Depending on how strongly the ontology is related to the database, the database concept may be above or below the domain concept. For example, if the database concept of the table "person" is above the domain concept of a "person", the relation to the database is strong. This indicates that according to this ontology nothing can be a person without also being a record in the database table of all persons. It also means that there may be records in the table that are not real persons. If the database concept is below the domain concept, the connection to the database is weak. A record from the database table about persons is to be considered a person; it inherits all qualities of a person. However, there may be persons who do not have records in the database.

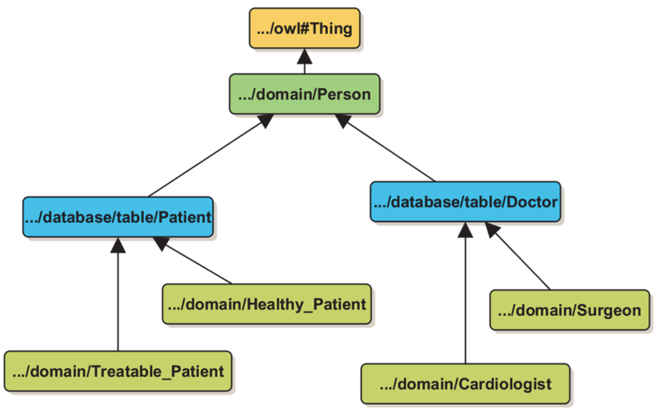
.../owl#Thing
.../domain/Person
.../database/table/Patient
.../database/table/Doctor
.../domain/Healthy_Patient
.../domain/Surgeon
.../domain/Treatable_Patient
.../domain/Cardiologist

Fig. 3. Mixed concept ontology fragment.

Figure 3 shows part of the class hierarchy of an example ontology built with mixed concepts. The blue class concepts represent database-connected classes. They are identifiable by the prefix they have. The green concepts are domain concepts. This example contains two different kinds of domain class concepts. The classes below the database concepts are strictly related to the database. This example ontology is constructed such that each example (instance) of a Cardiologist must also necessarily be a record in the database table "Doctor". This is implied by the class "Cardiologist" being below the class of all "Doctor" records. The domain class "Person" is more general. It combines records from two different database tables. The hierarchy also states that both patients and doctors are people. In this example, the class "Person" also leaves the possibility of other instances of the class "Person" existing, without being either a patient or a doctor.

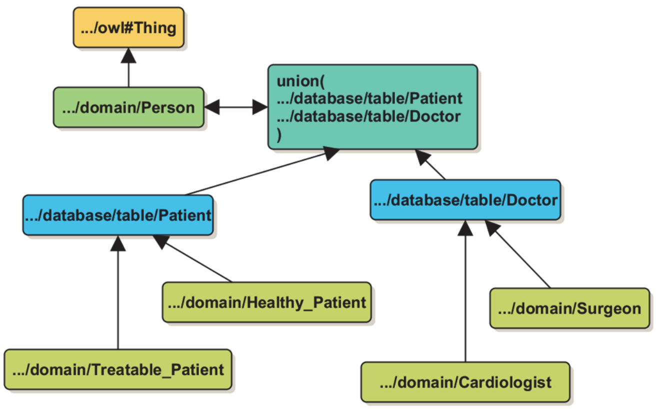
.../owl#Thing
.../domain/Person
union(
.../database/table/Patient
.../database/table/Doctor
)
.../database/table/Patient
.../database/table/Doctor
.../domain/Healthy_Patient
.../domain/Surgeon
.../domain/Treatable_Patient
.../domain/Cardiologist

Fig. 4. Stricter mixed concept ontology.

If there is a need for a concept that combines records from multiple tables but is also strongly related to the database, it may be defined using a union. For example, if it is necessary to state that both "Patient" and "Doctor" are people and also that instances of people can only be from these database tables, the ontology engineer can define a union of "Patient" and "Doctor" and equate the class "Person" to this union. Such an ontology can be seen in Fig. 4. Any union or other grouping concept consisting only of concepts related to a database is therefore strongly related to the database itself.

Column concepts are very similar. Instead of being added to the class hierarchy, they are added to the data property hierarchy. They may also have domain concepts describing data properties above and below them. During ontology building, these database data properties may be used for the definition of complex domain classes.

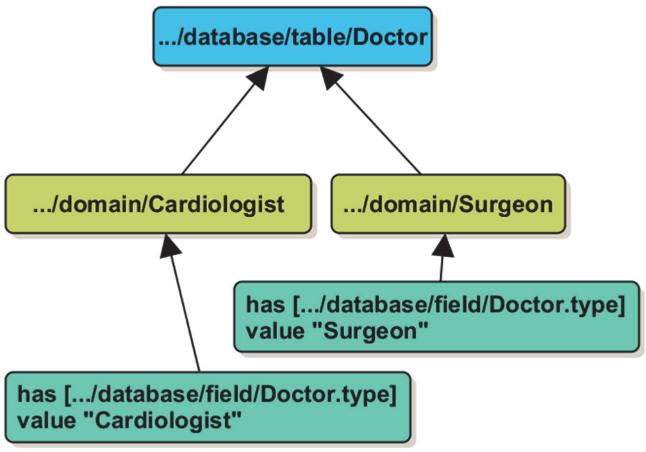
.../database/table/Doctor
.../domain/Cardiologist
.../domain/Surgeon
has [.../database/field/Doctor.type] value "Surgeon"
has [.../database/field/Doctor.type] value "Cardiologist"

Fig. 5. Complex database concepts.

Figure 5 shows an example of complex class definitions using a database column for criteria. When a complex class uses a database column of a certain table in its definition, it also should be a subclass of the concept of that table. This is because only individuals of this table class (database records) will have this particular column. In the example above, the complex classes provide sufficient criteria for the further classification of records as being cardiologists or surgeons. If there are multiple columns in different database tables, all having the same meaning, a domain data property may be introduced as a super concept to these data properties. Using such a super data property in the definition of a complex class would yield the same outcome, in this case.

Object properties are the only database concepts that require exceptional naming. Since they relate two tables, they must contain the names of both tables. The IRI of such object properties consists of a prefix indicating the ratio to a database, followed by the two database table names separated by a double dash. Such an IRI may look like this "…/database1/relation/Patient--Doctor".

The data retrieval engine can determine the case of a database table relation from the concept prefix. By splitting the object property concept name, using the double dash as a delimiter, and combining both parts with the common prefix, the IRI of both tables involved in this relation is obtained. This relation is to be considered directional. The relation goes from the first table to the second table. Any specifics about the nature of this relation are unfortunately lost.

The database concepts should be extended and related to as many domain concepts as possible. Any knowledge describing the domain must be added to the ontology. The database concepts should not be the most important part of any ontology. They may be used as a basis and a backbone of the ontology not only to map to data, but also to indicate what concept definitions are missing from a full ontology. However, database concepts that are not extended to more specific or more generic domain concepts serve very little purpose. The ontology should be richer and contain more knowledge than just the database structure.

V. Conclusion

The present paper proposed a novel approach to database to ontology mapping and ontology building. The presented approach provided an uncomplicated and comprehensible way to map information sources to domain concepts. Mapping is implemented in a way, which is more in line with the ontology itself. Instead of relying on an external tool, the ontology itself defines how the mapping impacts the concepts it contains. An external tool may rely on additional reasoning or calculation rules that have nothing to do with the capabilities and the methodology of the ontology. This means that, potentially, the ontology engineer should define the combined ontology using three different technologies. The engineer should consider the relational database, the ontology, and the rules of the mapping engine separately. In the proposed approach, the relations between database concepts and domain concepts are determined using standard ontology reasoning and do not require any additional tasks. A database concept will perpetuate its properties naturally within the concept hierarchy using the ontology internal reasoning. This approach is only capable of mapping directly to database objects. It does not have the expressiveness of other mapping approaches. Mappings defined in external files or in annotations may contain complex data queries. The degree to which this characteristic is assumed as a weakness is debatable. On the one hand, this approach seems to be lacking functionality. On the other hand, it is questionable if there should be complex queries contained in the ontology. If complex queries are needed, a view within the database may be created, and a database concept mapped to this view. Simplicity may be of value when creating a complex ontology. The ontology engineer should be concerned with the validity and usability of the ontology and not with complex data queries.

The use of the database concepts as an anchor point to a database can have a positive effect on ontology building. In the case when an ontology is built from the ground up, database concepts serve as a foundation for the domain concepts. The database concept is fleshed out with domain concepts related to them. The relations of domain concepts to database object concepts serve as indicators as to how well the ontology is. When domain concepts are not related to database concepts, these domain concepts may never have individuals. In this case, the question arises of how important these concepts are. When a database concept does not have any related domain concept, it simply describes the database structure without any additional information or knowledge. In this case, it is questionable if the ontology is a correct tool for describing database structures and whether the ontology should contain such a concept.

An ontology created using this approach is usable with any other ontology editor or tool. The proposed mapping approach has such little impact on the ontology and does not intrude on any existing ontology specifications or the methodology of ontology definition. Any additional tool may be used to work with such an ontology, without losing the mapping information contained in it.

References

[1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, Jun. 1993. https://doi.org/10.1006/knac.1993.1008

[2] M. G. Skjæveland, M. Giese, D. Hovland, E. H. Lian, and A. Waaler, "Engineering ontology-based access to real-world data sources," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 33, pp. 112–140, Aug. 2015. https://doi.org/10.1016/j.websem.2015.03.002

[3] E. Kharlamov, D. Hovland, M. G. Skjæveland, et al., "Ontology Based Data Access in Statoil," *Web Semantics: Science, Services and Agents on the World Wide Web*, Jul. 2017. https://doi.org/10.1016/j.websem.2017.05.005

[4] N. Konstantinou, D.-E. Spanos and N. Mitrou, "Ontology and database mapping: a survey of current implementations and future directions," *Journal of Web Engineering*, vol. 7, no. 1, pp. 1–24, March 2008.

[5] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao, "Ontop: Answering SPARQL queries over relational databases," *Semantic Web*, vol. 8, no. 3, pp. 471–487, Dec. 2016. https://doi.org/10.3233/sw-160217

[6] "MovieOntology scenario," Aug. 8, 2017. [Online]. Available: https://github.com/ontop/ontop/wiki/attachments/Example_MovieOntology/movieontology.obda. [Accessed: Sep. 10, 2017].

[7] G. Bumans, "Mapping between Relational Databases and OwL Ontologies: an example," *Computer Science and Information Technologies*, vol. 756, pp. 99–117, 2010.

[8] C. Martínez-Cruz, J. M. Noguera, and M. A. Vila, "Flexible queries on relational databases using fuzzy logic and ontologies," *Information Sciences*, vol. 366, pp. 150–164, Oct. 2016. https://doi.org/10.1016/j.ins.2016.05.022

[9] Y. An and T. Topaloglou, "Maintaining Semantic Mappings between Database Schemas and Ontologies," *Lecture Notes in Computer Science*, pp. 138–152. https://doi.org/10.1007/978-3-540-70960-2_8

[10] G. Mecca, G. Rull, D. Santoro, and E. Teniente, "Ontology-based mappings," *Data & Knowledge Engineering*, vol. 98, pp. 8–29, Jul. 2015. https://doi.org/10.1016/j.datak.2015.07.003

[11] C. D. C. Ta and T. P. Thi, "Improving the Algorithm for Mapping of OWL to Relational Database Schema," *Lecture Notes in Computer Science*, pp. 130–139, 2015. https://doi.org/10.1007/978-3-319-21024-7_9

[12] D. Ouyang, X. Cui, and Y. Ye, "Mapping integrity constraint ontology to relational databases," *The Journal of China Universities of Posts and Telecommunications*, vol. 17, no. 6, pp. 113–121, Dec. 2010. https://doi.org/10.1016/s1005-8885(09)60534-3

[13] K. Čerāns and G. Būmans, "RDB2OWL: A Language and Tool for Database to Ontology Mapping," In *Databases and Information Systems VI*, IOS Press, 2011, pp. 139–152. https://doi.org/10.3233/978-1-60750-688-1-139

[14] K. Čerāns and G. Būmans, "Database to Ontology Mapping Patterns in RDB2OWL Lite," *Databases and Information Systems*, pp. 35–49, 2016. https://doi.org/10.1007/978-3-319-40180-5_3

[15] M. A. G. Hazber, R. Li, X. Gu, and G. Xu, "Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology," *Applied Mathematics & Information Sciences*, vol. 10, no. 3, pp. 881–901, May 2016. https://doi.org/10.18576/amis/100307

[16] B. Motik, I. Horrocks, and U. Sattler, "Bridging the gap between OWL and relational databases," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 2, pp. 74–89, Apr. 2009. https://doi.org/10.1016/j.websem.2009.02.001

Henrihs Gorskis is a Researcher in the field of information technology at Riga Technical University (RTU). He received his *Mg. sc. ing.* degree in 2013. He is currently working on finalising his Doctoral Thesis. His research interests include data mining, ontology engineering, ontology-based database access and evolutionary computing and programming. He is especially fond of the Java programming language and uses it for both work and personal application development.
E-mail: henrihs.gorskis@rtu.lv

Ludmila Aleksejeva received her *Dr. sc. ing.* degree from Riga Technical University in 1998. She is an Associate Professor at the Department of Modelling and Simulation, Riga Technical University. Her research interests include decision making techniques and decision support system design principles, as well as data mining methods and tasks, and especially collaboration and cooperation of the mentioned techniques.
E-mail: ludmila.aleksejeva@rtu.lv

Inese Poļaka received her *Dr. sc. ing.* degree from Riga Technical University in 2014. She works as a Lecturer at the Institute of Information Technology of Riga Technical University (Latvia) and Leading Researcher at the Faculty of Medicine, University of Latvia. The main research interests include data mining, machine learning, classifiers, evolutionary algorithms and their applications, as well as bioinformatics and biostatistics.
E-mail: inese.polaka@rtu.lv