Ontology-driven relational database generation for heterogeneous data

Technical report

Christina Khnaisser[1,2], Luc Lavoie[1], Benoit Fraikin[1], Adrien Barton[1,3], Anita Burgun[2], Jean-François Ethier[1,2]

Khnaisser, C., Lavoie, L., Fraikin, B., Barton, A., Burgun, A., and Ethier, J.-F. 2019. Advanced ontology-driven relational database generation for heterogeneous data : Ontorelα. *Journal on Data Semantics (JODS) [Submitted 2019-09-20]*.

Abstract

A large volume of heavily fragmented data is generated every day in different contexts using various structures and different knowledge models. Because of this fragmentation and heterogeneity of databases, the integration of data of interest remains a major challenge. However, an adequate, efficient and shareable database can be built on the basis of a unified knowledge model. Recently, knowledge models have been developed using ontologies, whereas many kinds of data are still stored in relational databases. Still, designing a data model that captures the richness of the ontology is a challenging task. In order to facilitate such task, this paper presents a method for generating a RDB from an ontology, which reduces the gap between ontologies and relational database design. Moreover, a prototype named OntoRelα has been developed and evaluated to demonstrate the methods.
major challenge. However, an adequate, efficient and shareable database can be built on the basis of a unified knowledge model. Recently, knowledge models have been developed using ontologies, whereas many kinds of data are still stored in relational databases. Still, designing a data model that captures the richness of the ontology is a challenging task. In order to facilitate such task, this paper presents a method for generating a RDB from an ontology, which reduces the gap between ontologies and relational database design. Moreover, a prototype named OntoRelα has been developed and evaluated to demonstrate the methods.

* Jean-François Ethier : jean-francois.ethier@usherbrooke.ca
1 GRIIS, Université de Sherbrooke, Sherbrooke, Québec, Canada.
2 UMRS 1138 Équipe 22, Centre de recherche des cordeliers, Université Paris Descartes, Paris, France.
3 CNRS-IRIT, Toulouse, France.

1 Method

The method consists of generating a relational database (RDB) from an ontology. First, an ontology is analysed, then each ontological construct is converted into a relational database construct. The ontology is used to describe objectively a specific domain of discourse and the RDB is used to store the corresponding data in a structure that satisfies the integrity constrains derived from the ontology axioms.

General rules are defined in a way that the axiom semantic is preserved using the relational structure and adequate constraints. The ontological constructs are converted into relational constructs according to the rules described below and illustrated in figure 2.

Data type

A data type is converted to an SQL data type using a data type mapping.

The mapping must be configurable and a default one must be available.

Class

A class (other than Thing) is converted to a relvar including a database identifier attribute used as a primary key *dbid:dbid_type*. Each value of *dbid* uniquely identify an individual. The *dbid_type* must be configurable and a default one must be available (e.g. dbid_type:BIGINT, dbid_type:UUID).

Class inheritance axiom

A class inheritance axiom (isa operator) is converted into a referential key from the subclass relvar to the superclass relvar.

Class association axiom

A class association axiom is converted into an association relvar (a relvar that is related to two relvars or more) which has two attributes: the primary key of the domain class relvar and the primary key of the range class relvar.

One primary key composed of both attributes is defined.

Two referential keys are defined:

- one from the association relvar to the domain class relvar,
- one from the association relvar to the range class relvar,

One participation constraint is defined to check the number of individuals according to the participation specified in the axiom[1].

Finally, constraints are defined to mirror the reduction invariants generated during axioms' reduction.

Data type

A data type is converted to a relvar including two attributes: a data type identifier *dtid:dtid_type* attribute as a primary key and a data value *value:datatype* attribute.

The *dtid_type* must be configurable (e.g. SERIAL, UUID). Each value of *dtid* uniquely identifies a data type value. The data type relvar will contain exactly all the values of this data type used in the database, without repetition.

Data association axiom

A data association axiom is converted into an association relvar which has two attributes: the primary key of the domain class relvar and the primary key of the range datatype relvar.

One primary key composed of both attributes is defined.

Two referential keys are defined: one from the association relvar to the domain class relvar and the other from the association relvar to the range data types relvar.

One participation constraint is defined to check the number of individuals according to the

[1] Except, if this participation is [0..*].

participation specified in the axiom[2].

Finally, constraints are defined to mirror the reduction restrictions generated during axioms' reduction.

Annotation

An annotation is used to document the database and to provide multiple access interfaces in different languages using views.

A definition annotation is converted to SQL comment so they can be integrated into the DBMS catalogue (if the DBMS supports it).

A label annotation is converted to an attribute name or a view name (label view). For each label annotation, a label view is defined over the class table as follows:

- the name of the view is the class label,
- the heading of the view is the projection of the heading of the class relvar. The projection includes the renaming of the class relvar attributes according to their label annotation i.e. the "dbid" key is renamed "<view name> dbid".

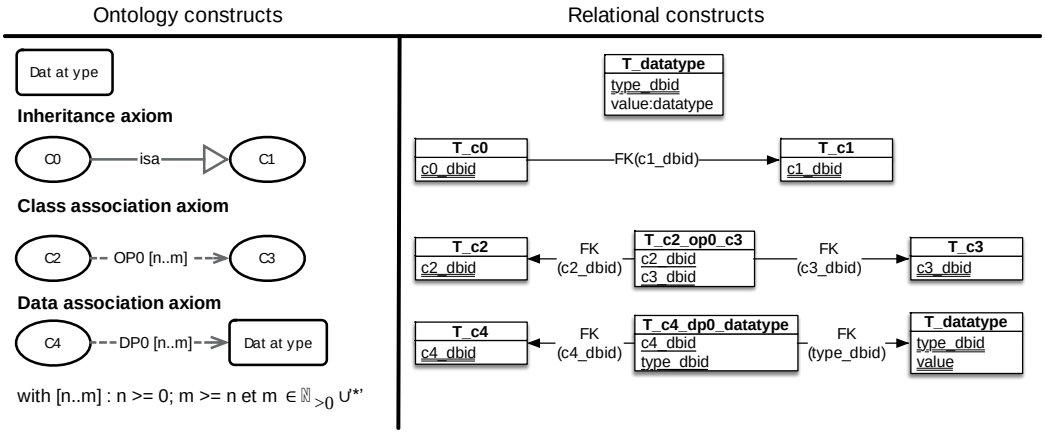

Fig. 1 Illustration of OntoRelα conversion rules.

2 Prototype

In order to illustrate the feasibility and the applicability of our method, a prototype, OntoRelα, was developed. OntoRelα generates from an OWL ontology and some configuration files: scripts for the RDB (RDB scripts), a list of warnings, a mapping dictionary (OntoRelDic) and a normalised ontology formalised according to μOnto. Figure 2 illustrates the inputs and outputs of OntoRelα.

[2] Except, if this participation is [0..*].

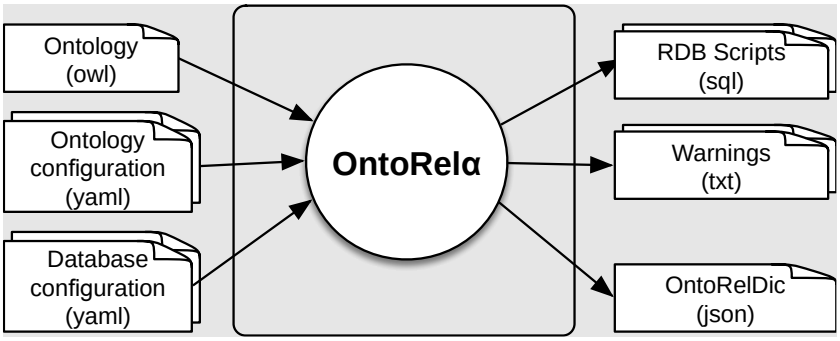

Fig. 2 OntoRelα context diagram.

OntoRelα is written in Java 8 with the following external libraries: OWLAPI 5.1[3] to load and analyse the ontology, JGraphT[4] to create graphs for the ontology and for the database, Snakeyaml [5] to analyse the configuration files, Jackson [6] to generate Json files, StringTemplate[7] to generate SQL scripts, and Logback[8] to log warnings.

Analysis process

The current version of OntoRelα accepts as input an ontology formalised according to OWL 2 (Motik et al. 2012b, a). The ontology configuration parameterises the analysis process. It consists of a list of classes and properties of interest and a list of OWL annotations properties.

The ontology verification generates a list of warnings which include the following:

- List of all classes and properties without labels or definition annotations.
- List of all classes without a data association axiom.
- Messages issued by the redundancy reduction step.

Conversion process

The conversion process converts ontology constructs into relational constructs using the RDB configuration file.

The RDB configuration parameterises the conversion process. It consists of the target RDBMS, the name of the database schema, the types of dbid and dtid attributes, the maximum length of table identifiers, data type conversion method, a list of natural languages of interest, and a list of correspondences between OWL types and SQL types according to the target RDBMS.

The RDB verification generates a list of warnings, including the following:

- A list of OWL data types without SQL correspondence.

[3] https://github.com/owlcs/owlapi
[4] https://jgrapht.org
[5] https://mvnrepository.com/artifact/org.yaml/snakeyaml
[6] https://github.com/FasterXML/jackson
[7] https://www.stringtemplate.org
[8] https://logback.qos.ch

- A list of truncated identifiers.

Finally, SQL scripts are generated. Currently, OntoRelα generates SQL for PostgreSQL 9+ but other dialects will be available.

The RDB is defined by a set of scripts including the following ones:

- The basic schema creation script includes creation statements for the base schema, types, relvars, candidate keys, and referential keys. A basic schema creation script includes a CREATE SCHEMA statement, for each datatype a CREATE DOMAIN statement and for each relvar a CREATE TABLE statement with a primary key and the referential key constraints.
- The function creation script includes creation statements for functions implementing the general constraints. A function creation script includes a CREATE FUNCTION statement for each general constraint: participation constraints, no-redundancy constraint, inheritance constraint, union constraint, and intersection constraint.
- The IRI schema creation script includes creation statements for the IRI schema and IRI views on all the relvars. An IRI view of a relvar is created by renaming the relvar and its attributes using the IRIs of the ontological construct. An IRI schema creation script includes a CREATE SCHEMA statement and for each relvar a CREATE VIEW statement using the short IRIs for the view and the attribute names.
- A LN schema creation script for each natural language of interest (ln). A LN schema creation script includes creation statements for the LN schema views on all the relvar. A label view of a relvar is created by renaming the relvar and its attributes using the label annotation of the language ln. A language schema creation script is generated for each natural language of interest. Each script includes a CREATE SCHEMA statement and for each relvar a CREATE VIEW statement using the label annotation for the view and the attribute names.

The IRI schema and LN schema are especially useful for querying by external applications.

2.1 Mapping dictionary

The mapping dictionary (OntoRelDic) describes the correspondence between classes, data types, axioms and tables. A class description includes IRI, the origin of the class (declared or generated by OntoRelα), the corresponding table (tableId), the English label (label), all related table generated for data association axioms (dataTables) and class association axioms (classTables). An association axiom description includes the axiom expression, the origin of the axiom, the corresponding table and all related tables.

For example, the following describes the class PDRO_0000024 and one of two of its axioms.


```
[…]
{
  "iri": "http://purl.obolibrary.org/obo/PDRO_0000024",
  "origin": "DECLARED",
  "tableId": "T0042",
  "label": "drug prescription",
  "dataTables": [],
  "classTables": [
    {
```

```
      "assocationTable": "T00e6",
      "dependentIri": "http://purl.obolibrary.org/obo/PDRO_0000007"
    },
    {
      "assocationTable": "T00e7",
      "dependentIri": "http://purl.obolibrary.org/obo/PDRO_0000003"
    },
    {
      "assocationTable": "T00e9",
      "dependentIri": "http://purl.obolibrary.org/obo/PDRO_0000005"
    }
  ]
}
[…]
{
  "subClassIri":"http://purl.obolibrary.org/obo/PDRO_0000024",
  "subClassTable":"T0042"
  "superClassIri":"http://purl.obolibrary.org/obo/PDRO_0000001",
  "superClassTable":"T0041"
}
[…]
{
  "expression": "PDRO_0000024 [1..1] :has_part PDRO_0000005",
  "origin": "DECLARED",
  "tableId": "T00e9",
  "associatedTables": ["T0019","T0042"]
}
[…]
```

2.2 Example

Here's an example from the Prescription of Drugs ontology PDRO. For clarity, a small subset of classes, properties, and axioms are described below using μOnto, and part of the generated RDB is illustrated with a relational diagram in Figure 7.

```
PREFIXE : "http://purl.obolibrary.org/obo/"

ONTOLOGY :PDRO
  LABEL @en 'Durg prescription ontology'

PROPERTY :has_part
  LABEL @en 'has part'

PROPERTY :has_value
  LABEL @en 'has value'

CLASS :PDRO_0000001
  LABEL @en 'Healthcare prescription'
  :has_part [1..*] :IAO_0000302
  :has_part [1..*] :PDRO_0000003
  :has_part [1..*] :PDRO_0000005

CLASS :IAO_0000302
  LABEL @en 'Author identification'
  :has_value [1..1] :String

CLASS :PDRO_0000003
  LABEL @en 'Patient identification'
  :has_value [1..1] :String

CLASS :PDRO_0000005
  LABEL @en 'Document creation time identification'
  :has_value [1..1] Date
```

```
CLASS :PDRO_0000024
  LABEL @en 'Drug prescription'
  isa :PDRO_0000001
  :has_part [1..1] :PDRO_0000003
  :has_part [1..1] :PDRO_0000005
  :has_part [1..*] :PDRO_0000007

CLASS :PDRO_0000007
  LABEL @en 'drug administration and dispensing specification'
  :has_part [1..*] :PDRO_0010022

CLASS :PDRO_0010022
  LABEL @en 'drug administration specification'
  :has_part [1..*] :PDRO_0000060
  :has_part [1..*] :PDRO_0000103

CLASS :PDRO_0000060
  LABEL @en 'drug product specification'
  :has_part [1..*] (:PDRO_0000044 OR :PDRO_0040002)
  :has_part [1..*] :PDRO_0000097

CLASS :PDRO_0000044
  LABEL @en 'drug product name'
  :has_value [1..1] :String

CLASS :PDRO_0040002
  LABEL @en 'active ingredient name'
  :has_value [1..1] :String

CLASS :PDRO_0000097
  LABEL @en 'drug identification number'
  :has_value [1..1] :String

 CLASS :PDRO_0000103
  LABEL @en 'prescribed dosing specification'
  :has_part [1..*] :PDRO_0000190

 CLASS :PDRO_0000190
  LABEL @en 'prescribed dosing specification'
  :has_value [1..1] :String
```

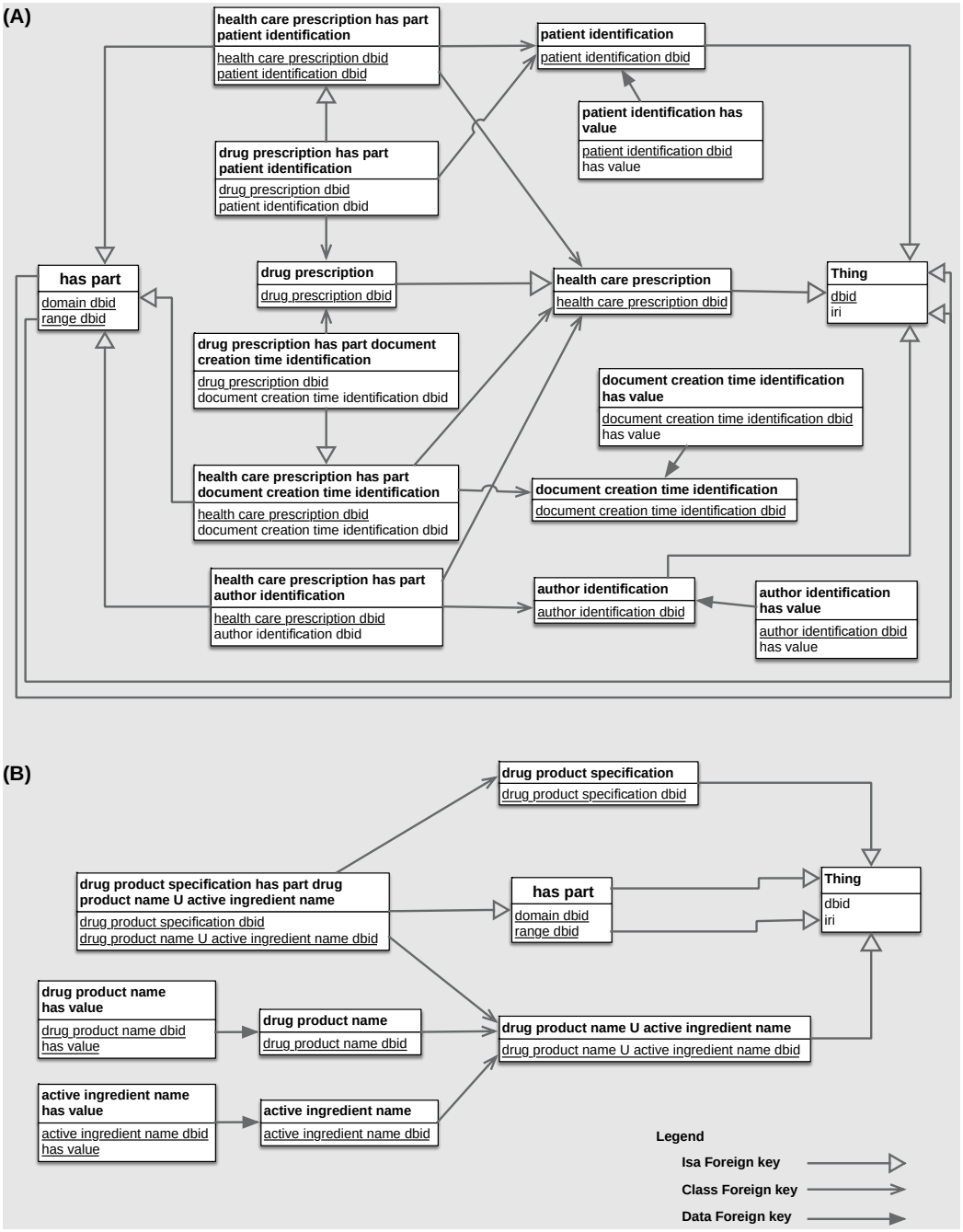

Fig. 3 Illustration of some part of the relational schema diagram.

3 Contributions and future work

To synthesise, our proposal differs from other existing methods in several ways (See Appendix A for more details):

- The normalisation of the database schema to allow a schema evolution that

only requires extensions or light modifications.

- The generation of advanced constraints such as participation, intersection and union constraints.
- The transformation of ontological annotations into SQL comments and views to provide documentation and multiple access points.
- The configurable transformation of OWL types into SQL types.
- The generation a mapping dictionary that enables structure reversibility.
- A prototype implementation, OntoRelα, as an open-source code and ready to be used with various OWL ontologies and PostgreSQL database.

The method handles more ontological constructs than existing methods. However, work is underway to cover the following:

- Converting equivalent axioms.
- Converting individuals into tuples and vice versa.
- Handling axioms with negation and complement operators.
- Defining rules for stored procedures and high-level relational view creation to facilitate data manipulation and querying.
- Optimise the conversion rules to reduce the number of tables.

Furthermore, a SQL generator for multiple RDBMS (Oracle, MSSQL, MariaDB, etc.) is under development.

4 Conclusion

This paper describes a new method for generating a RDB from an ontology and presents a functional implementation, OntoRelα. The results show, through examples, that the conversion rules are easy to apply and verify. Furthermore, OntoRelα demonstrates that ontologies can be a powerful tool to define a RDB by generating adequate structure and constraints that preserve the semantics of the converted ontological constructs. More results using various existing ontologies can be found on GitHub. Finally, we believe that the generated RDB can be used as a unified relational knowledge schema to semantically integrate existing heterogeneous data. Yet, many challenges and future works remain to handle more ontological constructs, to reduce the size of the generated relational construct and to offer an appropriate exploration tool for users of the generated RDB.

References

Motik B, Patel-Schneider, PF, Cuenca Grau B (2012a) OWL 2 Web Ontology Language Direct Semantics (Second Edition). In: W3C. https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/. Accessed 26 Jul 2018

Motik B, Patel-Schneider PF, Parsia B (2012b) OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). In: W3C. https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/. Accessed 3 Apr 2016