# Mapping Databases To Ontologies To Design And Maintain Data In A Semantic Web Environment

Olivier Curé
ISIS Laboratory, Office #240
Université de Marne-la-Vallée
Champs-sur-Marne, 77424, France

**Abstract.** This paper presents a global framework which enables the end-user to design, enrich and maintain an ontology from an existing database. These functionalities facilitate the development and maintenance of Semantic Web applications from frequently updated and domain-concerned databases. The efficiency of this framework is evaluated through the study of a medicine-oriented Semantic Web application which benefits from all the features of the DBOM framework.

**Keywords:** Ontology, Knowledge base, database, mapping.

## 1. INTRODUCTION

The Semantic Web is an extension of the current Web, considered syntactic, in which semantic markup languages enable computers to process and interpret Web resources. The backbone of this next generation web is the notion of ontology, understood in an information system way and computer science [7][8] rather than its philosophical definition. The World Wide Web Consortium (W3C) has recently released a set of recommendations, on Resource Description Framework (RDF) and Web Ontology Language (OWL), to tackle the design of ontologies for the Semantic Web.

This paper addresses two crucial issues for the development of Semantic Web applications. The first one is related to the design of ontologies, which is considered difficult to handle even though effective and usable tools are now available (Protégé[6]). The second one is concerned with data acquisition, another cumbersome task which faces the current Web's data overload. Most of these data are stored in databases and in unstructured text-based documents, usually generated from Word processors.

In [3], the authors emphasize that “databases are similar to knowledge bases because they are usually used to maintain models of some domain of discourse”. The DBOM (DataBase Ontology Mapping) framework exploits this proposition and deals with both ontology design and data acquisition issues via mapping with databases. This framework also proposes maintenance features for an OWL ontology considering updates of an existing relational database. A high potential of the DBOM framework is to design domain and application ontologies [8] from a domain-concerned database, meaning that an important part of the data available for this domain are contained and well described in the database. A valuable example in the context of XIMSA (see section 4) is an exhaustive database of all OTC (Over-The-Counter) drugs available in France. An important fact about such databases are that update operations may also update the domain. Thus the ontology requires to be synchronized with the database. The proposed framework offers such features and extends them to permit symmetrical maintenance solutions, meaning that controls and maintenance operations are done both ways : from the database to the ontology (ontology updating, e.g. adding new instances, etc.) and from the ontology to the database (e.g. consistency checking).
This paper is organized as follows :

- section 2 highlights the motivation behind the DBOM framework based on studies of available solutions in this field.
- Section 3 proposes an overview of the DBOM framework which describes mapping, enrichment and maintenance issues.
- section 4 proposes a full-fledged example based on DBOM : the XIMSA (eXtended Interactive Multimedia System for Auto-medication) project.

## 2. MOTIVATION

The potential collaboration between database schemata and ontologies is an active field, especially since the emergence of the Semantic Web. A SWAD-Europe deliverable [12] emphasizes that many implementations (D2R Map [2], KAON Reverse [11], etc.) are already tackling this issue for RDFS and OWL ontology languages. These solutions help to design an ontology from a database schema and to instantiate classes from tuples. A study of [12] highlights the absence of a global framework which enables the maintenance of ontologies from databases. These functionalities are quite important for Semantic Web applications where TBox and Abox (respectively terminological and assertional knowledge) states of the knowledge base must be synchronized with the current domain state. Available solutions enable the updating of ontologies by recreating, from scratch, all the instances from the database via a mapping process. Such an approach may be difficult to handle with large ontologies (up to few mega bytes) and frequently updated databases. Thus making scalability an important issue. DBOM satisfies this issue by providing a dynamic maintenance solution between the database and the ontology.

DBOM focuses on the development of web

ontologies used in the context of Semantic Web applications. A major constraint for such framework is that a fully automatic solution to convert data from a relational schema into a Description Logic knowledge base schema (with TBox and ABox) can never prove to be fully satisfactory. A reason is that the relational schemata are usually the only source of information as far as (Extended) Entity-Relation diagram, richer in terms of semantics (cardinality constraints, etc.) are not available. Thus the solution needs to be semi-automatic, meaning that the DBOM end-user/ ontology designer (referred as the designer in the rest of this paper) builds the database to ontology mapping file. The designer is then the source of information which permits to map the semantics less database schema to a richly axiomatized ontology. Once validated, the mapping file is automatically processed.

The approach adopted by DBOM is that only a portion of the database tables and attributes necessarily map to the ontology. Nevertheless, the designer still has the freedom to map as many relations and attributes as he wants to the ontology. But in order to ensure fast navigation and effective reasoning activities, the design of ontologies provided with the minimum of classes and properties is preferable. For example, the design of the XIMSA ontology (see section 4) does not incorporate attributes such as drug price because this field is not useful in the current XIMSA reasoning context. Anyhow, such information are valuable to end-users and are still accessible via bridges, designed within DBOM, between the database and the ontology.

The vision of the DBOM framework is to develop applications that use the ontology for inference purposes and is able to bind the inference results to the database thus providing exhaustive information to the end-user. In order to reach this goal, the designer of the mapping must be aware of the database schema and needs a clear vision of the characteristics of the Semantic Web application about to be implemented.

The main motivation behind the maintenance features remains in the database / ontology separation. This separation requires that the database schema is not modified during mapping processing and enables users of the DBOM framework to benefit from :

- database features which are not available in ontology engineering. We can emphasize mechanisms such as concurrency control, transaction, crash recovery, advanced storage techniques and query languages.
- OWL ontologies, and underlying Description Logics, reasoning tools, e.g. Racer [9], which enable to check the consistency of the ontology.

## 3. DESCRIPTION OF COMPONENTS

The DBOM framework is implemented in Java and is based on Hewlett Packard's Semantic Web Jena API [10]. Jena has been selected because it is an efficient open-source framework popular in the Semantic Web developer community. Among interesting features, Jena provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine.

The description of the DBOM framework is divided into four distinct components : design, enrichment, ontology and database maintenances. A global vision of the framework is proposed in Figure 1.

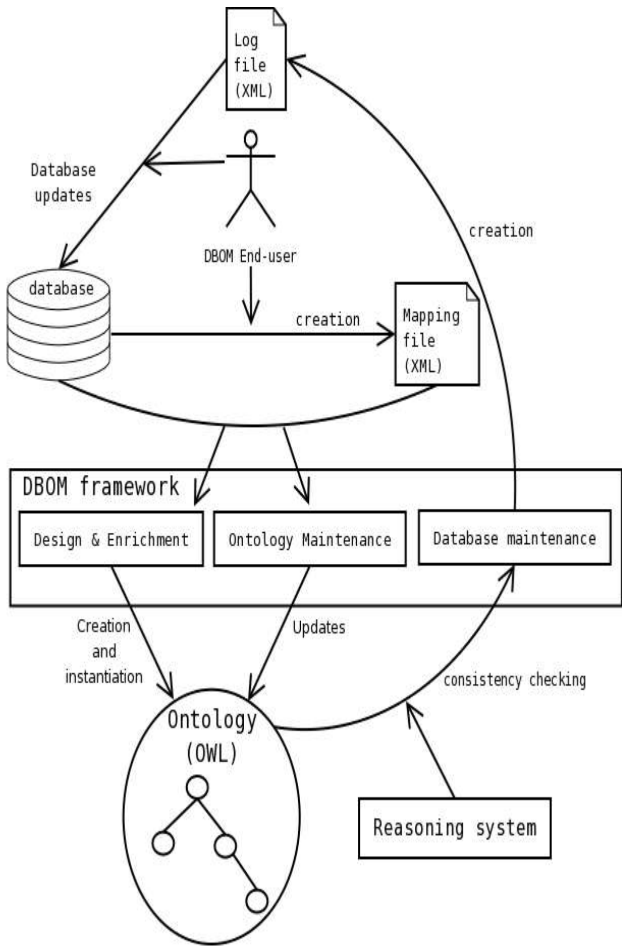

**Fig. 1** Global vision of interactions with the DBOM framework.

### 3.1 Ontology design

DBOM's ontology design implementation respects two of the main prerequisites described in section 2 :

- a semi-automatic solution is adopted. The ontology designer uses his knowledge of the database schema to design the mapping to the ontology. The mapping is supported by an XML file whose parsing implies to validate an XML Schema. Thus the designer needs expertise in DBMS, XML and ontology modeling.
- the existing database schema is not changed. Thus actions effectuated on the database are data manipulation operations of the SQL language (DML). This approach is adopted because the designer has skills in using DBMS, thus writing SQL queries should not be a difficult task. Storing SQL queries directly in the XML mapping file implies improvements in terms of readability and usability. The ability to easily create individuals from

instance data spread over several tables is another advantage provided by the use of SQL queries.

The first step a designer has to describe in the XML mapping file is to provide the information related to the DBMS connection. DBOM enables connection with RDBMSes offering JDBC and ODBC access. Following this section namespaces definitions can be declared. Then follows the heart of the ontology design with the creation of concepts and properties.

The ontology design phase consists in describing the relations, attributes and possible conditions implied in the design of the ontology classes and properties (data and object properties). The mapping addresses the design of an ontology according to constructors of the OWL Lite and DL languages. This approach enables DBOM to generate the TBox with class and property hierarchies as well as restrictions. The assumption of acyclicity is common in Description Logic knowledge bases [3] and is the responsibility of the designer.

The mapping must conform to a set of rules :

- class individuals follow the format : className_xxx. The className is provided in the mapping file by the designer. The xxx value is equal to the primary key value of the corresponding database relation. DBOM allows the creation and manipulation of compound identifiers. This rule satisfies the correspondence between instances of the ontology and tuples of the database.
- for ontology readability, class individuals must be provided with text attributes stored in a rdfs:label element and/or datatype properties defined in the mapping file

The purpose of the rest of this section is to emphasize the main constructions possibilities of the mapping. For readability reasons, the code fragments are kept to their minimal and essential forms. It is then necessary to stress that functionality definitions are not provided for namespaces, abstract classes, etc.. An examination of the XML Schema [5] presents all the functionalities of the mapping language. In order to provide a concrete example, the description of the mapping is based on the following simple relational schema :

```
     gender(idGender,valueGender)
     person(idPerson, lastNamePerson,
firstNamePerson,    zipcodePerson,idGender#)
     parent(idChildPerson#, idParentPerson#)
```

The definition of the Person ontology concept is described with the following XML code fragment :

```
<dbom:class dbom:name="Person"
dbom:src=”person”>
      <dbom:sqlstatement query="SELECT
person.idPerson,
person.lasteNameValue,person.firstNameValue
      FROM person;"/>
      <dbom:bind>
       <dbom:id>
         <dbom:value>1</dbom:value>
       </dbom:id>
       <dbom:label>
         <dbom:value>2</dbom:value>
         <dbom:value>3</dbom:value>
       </dbom:label>
      </dbom:bind>
  </dbom:class>
```

The dbom:class element maps to the creation of an OWL concept named after the value of the dbo:name attribute, (in this example, Person). An equivalent dbo:subclass element enables to design class hierarchies from SQL statements and requires the name of the extended class.

The design of richly axiomatized concept is also possible with the DBOM design mapping. Lets suppose we would like to design a Male concept corresponding to male persons from the Person relation. In order to store the corresponding ontology restrictions, DBOM proposes constructors such as onProperty and hasValue. SomeValuesFrom and allValuesFrom, as well as all OWL Lite and DL languages constructs are also accessible via the mapping file. The following example provides the definition of the Man concept as being a person with a 'male' value for the object property 'hasGender' :

```
  <dbom:definition dbom:className="Man"
dbom:defType="equivalentClass">
    <dbom:conjunct>
       <dbom:element
dbom:className="Person"/>
       <dbom:element
dbom:type="hasValueRestriction"
dbom:objPropertyName="hasGe
nder" dbom:value="Gender_1"/>
    </dbom:conjunct>
  </dbom:definition>
```

DBOM mapping features enable end-users to defined all kinds of properties :

- data type properties are easily integrated within ontologies according to a mapping between relation attributes and the range of this property. In the previous example, we can assign a hasZipCode property to the Person class.
- All kinds of object properties (functional, transitive, symmetrical, etc.) can be defined with or without property restrictions. In the previous example, a hasParent object property may be defined with the class Person as domain and range. This property may be declared transitive to enable the recursive task to access ancestors.

Property hierarchies features are also present in the mapping solutions. With the previous example, we can define a hasFather property, a sub-property of the hasParent property. The corresponding SQL statement would be :

```
SELECT person.idPerson,
person.lastNamePerson,
person.firstNamePerson FROM person, gender
             WHERE
person.idGender=gender.idGender
       AND valueGender like 'male';
```

### 3.2 Ontology enrichment

The performances of the enrichment phase are increased due to the storage of SQL queries in the mapping file. The query elements (attributes in the SELECT clause, table names in the FROM clause, etc.) are not decomposed in the XML file thus a SQL query generation is not necessary. A simple parsing of the mapping files enables query execution in the database.

The SQL statements in the DBOM mapping file support the creation of class instances and assign values to instance properties. In the previous Person concept example, the dbom:value element specifies the field number in the corresponding SQL query which is attached to the nesting element. In this article example, the idPerson column (first in the list) maps to a Person concept rdf:ID element. The following Person table tuple :

```
(101, 'Cure', 'Olivier',77500, 1)
```

will generate the following individual :

```
<person rdf:ID="person_101">
  <rdfs:label>Cure Olivier</rdfs:label>
</person>
```

The enrichment implementation is considerably facilitated by the use of the Jena API which proposes classes and methods to generate OWL Lite and DL ontologies in various formats (N3, N-TRIPLES, RDF/XML). The order in which the Jena methods are called is important and follows respectively the TBox creation, with datatype properties, classes and object properties, and the ABox creation with concept instantiations and instance property assignments.

### 3.3 Ontology maintenance

The maintenance phase is concerned with (non-schema) updates of the databases which also need to be effectuated on the ontology. The enrichment phase is usually performed once for an ontology schema while maintenance may be processed several times.

The ontology maintenance mechanisms is centered on the fact that database updates may imply ontology updates. This possibility is motivated by the presence or absence of the updated attribute in the ontology. The implementation solution is provided by the adoption of SQL triggers and external to the database Java methods.

Triggers offer a facility to autonomously react to database events by evaluating a data-dependent condition and by executing an action whenever the condition is satisfied. The DBOM framework makes an extensive use of the Event-Condition-Action (ECA) trigger model. The maintenance module necessitates triggers for all three events (Insert, Update and Delete) for all relations involved in the ontology mapping. Considering large ontologies, this fact may involve the creation of an important number of triggers. It is known that the design of triggers by hand is a difficult and error-prone task. DBOM proposes to generate automatically all triggers given the assumption that the database schema assigns ON DELETE and ON UPDATE clauses for all relations when required by primary/foreign key binding. This solution offers the opportunity to fire triggers, and maintain the ontology, using a cascading approach. The Figure 2 presents the general policy for ontology concept triggers generation :

```
for each concrete class definition in the mapping file
{
    generate an INSERT trigger for the involved relation with restriction in the WHEN clause if required.
    generate a DELETE trigger for the involved relation with restriction in the WHEN clause if required.
    for each attribute in the SELECT clause {
        generate an UPDATE trigger for that attribute for the involved relation with restriction in the WHEN clause if required.
    }
}
```

**Fig .2** Algorithm for concept triggers.

We omit the trigger generation algorithms related to properties which we consider straight forward to the policy provided with the algorithm in Figure 2.

All the generated triggers are characterized by being:
- Fired after events occurrence.
- Executed at the row level, meaning that the rule is triggered separately for each tuple.
- Related to actions which are automatically executed, external generic programs, written in Java.

The generation of triggers involves an XML parsing and pattern matching operations on the SQL queries. Performance figures for this task can be improved with a decomposition of SQL queries elements but such an approach is done at the price of usability and readability of the mapping file (see Section 3.1 and 3.2). The next version of DBOM will be provided with a QBE-like (Query By Example) solution for the mapping file creation. This approach will then tackle the best of both worlds : fast access to decomposed query elements and usability considerations.

### 3.4 Database maintenance

The last component of DBOM is related to the database maintenance which is ensured by consistency checking of an ABox w.r.t. a TBox. This database maintenance is executed after an effective, at least one trigger has been fired, ontology maintenance. The objective of this maintenance is to ensure that new instances and property values are consistent with the semantics of the ontology, something the DBMS can not process due to its lack of detailed semantics.

DBOM uses Racer [9] because it proposes the following characteristics :
- it is the only publicly available DL system

that supports full ABox reasoning for an expressive DL,

- it enables a Java application to communicate with a DL reasoner via the DIG ([1]) interface, thus proposing an HTTP-based standard for connecting our client program.

The current philosophy of the database maintenance is not to act directly on the database Thus the approach adopted is to propose a log file, in a XML format, to the end-user. This log file presents the inconsistencies detected by Racer providing the following information : date of detection of inconsistency and identifier of the inconsistent individual.

## 4. THE XIMSA EXAMPLE

The main features (design, enrichment and maintenance) of the DBOM framework are used on the XIMSA system [4]. The XIMSA web application aims at providing information and services to the general public on mild clinical signs and related treatments and medications. XIMSA is supported by Semantic Web technologies such as an ontology written in OWL DL. The auto-medication ontology has been designed using DBOM based on a drug database which contains all drugs available in France. For each drug, the database regroups all the data of the Summary of Product Characteristics (SPC) plus extra information such as opinions from health care professionals and a drug rating.

The design of the ontology is the result of a collaboration with the clinical pharmacology department at the Cochin hospital in Paris. DBOM has proved to be valuable during this collaboration because :

- it helps all actors of the design (health care professionals and ontology designer) to concentrate solely on the conceptual tasks,
- the ontology becomes a concrete, useful entity, instantiated with thousands of objects, within seconds after the conceptualization.

At the time of writing this paper, the XIMSA ontology incorporates concepts and properties concerned with the field of drug auto-medication specialized in detecting drug contra-indications and patient allergies. Future version of XIMSA will imply to extend the ontology with new concepts and properties. DBOM functionalities will prove useful during this extension phase because features addition are done at the design level and are automatically processed at the enrichment and maintenance levels.

The integration of the ontology in XIMSA enables the patient to control drug prescription regarding his Simplified Electronic Health Record (SEHR). The SEHR is created and maintained within the web-based XIMSA interface and enables the patient to store general (name, gender, date of birth, etc.) and health related (clinical and drug interactions, drugs being prescribed, etc.) information. An accurate and up-to-date SEHR assists the system in providing safe drug prescription to a specific patient. A ruled-based mechanism handles the ontology, the SEHR and a particular request of the end-user (for example requiring an antitussive drug) to propose hyper links of safe to prescribe drugs. The result of the inference mechanism provides identifiers of eligible ontology instances. Due to the correspondence between the ontology instances and the database tuples, the end-user can click on a hyper link and obtain all the information stored in the database about this drug, including information not contained in the ontology (e.g. drug price).

An interesting fact about the XIMSA system is the omnipresence of the drug database. Retrieval queries are involved in the SEHR creation and maintenance as well as during the drug proposition. A secured administrator page is also provided to maintain the database. This part of the application is bound to transaction, concurrency control and crash recovery issues. Such a system is tremendously gaining, in terms of functionality and usability, from the database / ontology separation and collaboration.

Finally, updates of the ontology produce a consistency checking. Lets consider the addition of a new drug in the database where the therapeutic class is not consistent with the Recommended International Non-proprietary Name (RINN – the active molecule) of the drug. Although the statement recorded in the database is valid, its semantic is wrong. The consistency checking of the ontology will report, in a log file, that an inconsistent statement has been added in the database.

## 5. CONCLUSION

The DBOM framework enables the design and maintenance of high quality ontologies by providing correctness, minimally redundant data and richly axiomatized descriptions characteristics. The correctness quality is provided by the capture of the intuitions of domain experts which is facilitated by a conceptual-concerned collaboration with the designer. This collaboration also benefits from a knowledge representation language abstraction and fast access to a realistic, richly instantiated ABox. The minimal redundancy quality is provided by the database / ontology separation considering that the designer respects the DBOM philosophy : relations and attributes not concerned with inference mechanisms should not appear in the ontology and stay solely in the database. The richly axiomatized quality implies that descriptions are “sufficiently” detailed. DBOM satisfies this requisite in providing the ability to add restrictions during concept and property constructions.

This quality achievement and the ability to propose a knowledge base fast prototyping tool emphasizes that DBOM can be useful for the development of Semantic Web applications. XIMSA remains a valuable experience on the development of such elementary yet useful applications. The dual, synchronized maintenance of the database and the ontology highlights the importance of such feature in a knowledge-based environment.

Although the XIMSA auto-medication ontology contains more than 6000 drug products, 1500 RINN and 500 therapeutic classes ; we believe that studies with larger ontologies, meaning larger TBoxes and

ABoxes, need to be conducted. Performance surveys should also be conducted with such ontologies.

The DBOM framework also requires the implementation of a graphical QBE-like solution for the design of database to ontology mapping file.

## 6. REFERENCES

[1]Bechhofer, S. : The DIG description logic interface: DIG/1.1. In Proceedings of the 2003 Description Logic Workshop (DL 2003), 2003.

[2]Bizer, C. : D2R MAP - A Database to RDF Mapping Language . The Twelfth International World Wide Web Conference (WWW2003), Budapest, Hungary, May 2003. Available on line at :http://www.bizer.de/ .

[3]Bordiga, M. ; Lenzerini, M. ; Rosati, R. : Description logics for data bases. Chapter 16 in The description logic handbook, editer by Baader, F. ; Calvanase , D. ; McGuinness, D. ; Nardi, D. ; Patel-Schneider, P. Cambridge University Press.

[4]Cure, O. : XIMSA : eXtended Interactive Multimedia System for auto-medication. To appear in IEEE CBMS 2004 (Computer-Based Medical Systems). Bethesda, USA. June 2004.

[5]DBOM : Data Base Ontology Mapping home page at : http://www.univ-mlv.fr/~ocure/dbom.html/.

[6]Gennari, J. ; Musen, M. ; Fergerson, R. ; Grosso, W. ; Crubezy, M. ; Eriksson, H. ; Noy, N. ; Tu, S. : The evolution of protégé: an environment for knowledge-based systems development. International Journal of Human-Computer Studies, 58:89 123, 2003.

[7] Gruber, T. : A translation Approach to portable ontology specifications. Knowledge Acquisition. Vol. 5. 1993. 199-220.

[8] Guarino, N. : “Formal Ontology and Information Systems''. In the Proceedings of FOIS 1998. Also in Frontiers in Artificial Intelligence and Applications, IOS-Press, Washington, DC, 1998.

[9]Haarslev, V.; Moller, R. : Racer: A Core Inference Engine for the Semantic Web. In the proccedings of EON 2003 (Evaluation of Ontology-based Tools). Proceedings available online at http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-87/.

[10] Jena framework home page http://jena.sourceforge.net/

[11] Karlsruhe Ontology projet web page at : http://km.aifb.uni-karlsruhe.de/kaon2/frontpage .

[12] SWAD-Europe Deliverable 10.2: Mapping Semantic Web Data with RDBMSes. Available online at : http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/ .