International Conference on Control, Engineering & Information Technology (CEIT'13)
Proceedings Engineering & Technology - Vol.1, pp. 103-108, 2013

Ontology Maps to Database: OntoMapDB approach

Nakhla Zina
Computer science department. BESTMOD Laboratory,
High Institute of Management, University of Tunis,
41, City Bouchoucha, 2000 Bardo, Tunis, Tunisia
zinanakhla@yahoo.fr

Nouira Kaouther
Computer science department. BESTMOD Laboratory,
High Institute of Management, University of Tunis,
41, City Bouchoucha, 2000 Bardo, Tunis, Tunisia
Kaouthar.Nouira@isg.rnu.tn

***Abstract*—Ontology aims to organize knowledge of specific domain and provides a definition for the vocabularies and their relationships. Ontology plays an important role in the construction of database, it can be used to extend the classical model of databases. Several approaches were proposed to build database based on ontology structure. This Database called 'Ontology Based Data Base' (OBDB) stores ontology and instances, which is more adaptable for computer application because it insures the interoperability between computer application and the OBDB. The OBDB approches presented in the literature has limits as focusing on specific type of ontology. In this paper, we propose a new approach of OBDB representation called 'Ontology Maps to DataBase (OntoMapDB)'. Our approach aims to manage information in database from ontology source, facilitate access to data for computer application and help developer in conceptual step. It consists on mapping ontology components as concepts, instances, and properties to database components as table, records, and attributes. OntoMapDB treats all types of ontology, canonical and non canonical ontology. It does not depend on a specific ontology model and it is relevant for each domain.**

***Index Terms*—Ontology, Ontology based database, Database, Mapping, Design.**

I. Introduction

The information is stored in different source, which is often heterogeneous. The information source must contain data needed for a given domain and must be efficient and sharing. In order to establish efficient information sharing, it is necessary to find a suitable information source.

Several works related to DataBases (DB) in the last decade have focused on increasing information representation. Many representation models are proposed as the relational model, DB deductive model, and object-relational model. In addition, many work present the notion of ontology as a conceptual model of shared domain [1] [2], which describe all information in a specific domain in a consensual manner to conceptualize application areas. Ontology groups and structures volume of information. It aimed to group and structure specific domain concepts and the relationship between them [3]. The amount of data defined by reference ontologies has increased considerably. Ontology is used to manage large volumes of data and it is a solution to limit domain.

Ontology used to express the different conceptual models corresponding to different computer application and to provide unambiguous definition for terms used in a software system [4]. The explicit representation of ontology in each DB facilitate the integration of data and the generation of DB model in conceptualization step. Ontology plays an essential role in information integration. We can use ontology as a support to the computer application, mainly in the creation of a DB.

The Design of DB is a difficult step, because developer must first acquire the domain of application and identify the components needed to develop DB. DBs are application oriented and we cannot have a general DB for many applications. Consequently, conceptualization of DB model takes time and effort and the obtained result is non standard. To solve this problem, approaches have been made to allow the persistence of ontologies and their instances in a database called Ontology-Based DataBase(OBDB), which aimed to construct a DB based on the structure of ontology. OBDB [5] [6] is a data source that contains ontology, a set of data, links between them and the elements that define the ontological sense. It aims to sharing, management, integration and querying structured data associated with large conceptual ontologies.

In this paper, we discuss a new approach to construct database model from ontology. Our approach aims to facilitate the exploration of information stored in the ontology, facilitate the DB design, assist the DB designer, provide DB standardization, particularly in the health context. Our approach is different from other, because we propose a flexible and simple method, which is relevant for each type of ontology and for each domain, also it produces a simple and clear DB model.

The paper is structured as follows: In section 2, we present different types of ontology and their components. In section 3, we present and, we discuss the approaches of OBDB mapping in the literature. In section 4, we present OntoMapDB approach. In section 5, we discuss the mapping process of OntoMapDB approach. Then, we evaluate approach by applying in a cardiac ADE ontology and heart failure ontology. Finally, we conclude with a summary and perspectives.

II. Ontology

Researchers need to share a common structure of information in a different domain. These structure contain readable definition of basic concepts of a particular area and their relationships. An ontology is a formal system consists on a set of important concepts characterized domain, explicit and shared among different users whose goal is to represent the knowledge of a specific area.

According to Gruber [2], 'an ontology is an explicit specification of a conceptualization'. Studer [7] defines ontology 'as an explicit formal specification and a shared conceptualization'. We can explain the definition of ontology as follows:

- Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon.
- Explicit means that the type of concepts used, and the constraints on their use are explicitly .
- Formal refers to the fact that the ontology should be machine-readable.
- Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.

In general, an ontology provides the means to express the concepts of a field in hierarchical organizations and defining their semantic properties in the language of formal knowledge representation [8]. Ontology provides a common structure of information relative to specific domains. The ontology makes domain unambiguous and provide a common understanding, which can be communicated between people and application systems. Ontologies present the knowledge of a particular field. This knowledge is formalized as the following components, concepts, relations, axioms and instances [9]:

- Concept or class: is a type of object in the universe. It can represent an object, idea, or an abstract concept. A concept can be divided into three parts: a term, a notion and a set of objects [9].
  - The term of a concept is the linguistic expression commonly used to refer to it.
  - The notion refers to the design intent. It contains its semantics is defined using properties (attributes and relations), rules and constraints.
  - The set of objects defined by the concept form what is called the extension of the concept. These are the objects to which the concept refers (its instance).

  In summary, a concept represents all descriptions of a task, object, function, action, strategy, process of reasoning, idea etc ...
- Relations: a type of interaction between the concepts of a domain. They reflect existing associations. Ontologies usually contain only binary relations [10].
- Axioms: to model claims they considered obvious and had no need of evidence, about the abstractions of the domain resulted in the ontology. The formal axioms are used to verify the consistency of the ontology [10].
- Instances: are used to represent items in a field. They are the extensional definition of concepts of the ontology [10].

Ontology can be classified into two types [11]:

- Canonical ontology: that contains only primitive concepts. The primitive concepts can define the boundaries of the area conceptualized by an ontology [12].
- Non canonical ontology: contains primitive concepts and defined concepts. These concepts extend the vocabulary of ontology. Many ontologies models support the defined concepts as a description logics, Frame Logic (F-Logic), Web Ontology Language (OWL)... [13]

III. Related Works

OBDB is defined as "'a database model that store both ontologies and their instances in a single database"' [14]. The OBDB represents ontologies, data structure and links between data. It brings new characteristics, that we cannot find in the traditional DB. It stores both the ontology and their instances in the same repository and it presents many storage layouts for storing ontologies and data (vertical, horizontal, binary), where each representation has its advantages and limitations. The ontology referencing OBDB instances may be expressed in various formalisms (Resource Description Framework (RDF), OWL, etc.) [15].

Many OBDB model have been proposed in the literature, which differ following, model ontologies supported, database schema used to store ontologies represented in this model, database schema used to represent the instance of ontology, mechanisms used to define the link between instances and concepts.

Three main approaches have been followed to represent ontologies in a database [11]: Single table approach, dual schemes approach, and table per class approach.

A. Approach 1: Single table approach

This model is independant of ontology structure. It's based on storing ontologies elements such as classes and properties in a single table with three columns (subject, predicate, object). This triplet used to characterize each class by a name, a comment and its superclass [16] . The vertical method is used, to represent instances (each instance is defined by a triplet). It used to characterize each instance by a class name, and the property which describes instance.

The main weakness of this approach is the non-distinction between ontology and data engender a large number of joins in the table [17]. To avoid this problem, new system have proposed which separate between ontology and data.

B. Approach 2:Dual schemes approach

The approach uses two tables to store each concept model in the ontology and instance [17] [18]. For instance level, three schemes have been proposed to represent instance:

- A vertical table can be used to store the instance level as triplets [17].
- Binary representation each class is represented by an unary table and each property by a binary table. The binary approach is followed by Sesame and RDF-Suite [17].
- Another approach, called table per class, associates to each class a table of authorities which each column corresponds to a property [19].

C. Approach 3: Table per class approach

The approach depends on ontology structure. The ontologies are represented in a pattern depending on the model used ontology (RDFS, OWL, etc.). The representation of data in this approach called horizontal representation, which represented instance in a relational table. The table has a column for each property used to describe instance of this class [11].

Ontology DataBase (OntoDB), which is a representation model of OBDB based on table per class approach. It proposes a structure for storing and evolve the ontology model. It is composed on four parts, the two first parts data and meta database, which stores the instance data and the diagram. The second part represents ontology, and another part, called meta-schema that represents instances. OntoDB model of OBDB is based on PLIB ontology model.

Consequently OntoDB consists on many tables, which requires many joins in queries. Also, the large number of joins between tables engendered a delays in query execution. OntoDB architecture is based on canonical ontologies. This approach is not flexible, it not integrates many kinds of ontologies [6] as non canonical one.

A set of query are used to evaluate the performance of model and show that approach 3 is more performed than approach 1 and approach 2 [20] [18].

IV. OntoMapDB: Ontology Maps to DataBase

Our work consists on creating a new method for DB construction from ontology. The mapping from ontology called ”Ontology Maps to DataBase (OntoMapDB)” to DB can be defined as a set of correspondence between DB’s tables, columns, primary and foreign keys, attributes... etc. and the ontology’s concepts, relations, properties...etc. Our objective is to facilitate the extraction and the use of information in the computer application by the construction of an OBDB. OntoMapDB stores ontology information and structures in a DB. It helps developer to construct DB. OntoMapDB is different to other approaches because it supports multiple models of ontologies (canonical ontologies and non canonical ontologies). The main steps of OntoMapDB are:

- Defines how to generate a DB, based on concept and instance stored in ontology.
- Defines table in DB.
- Defines column for each table in DB.
- Defines relationship between table in the DB.

The method describes four types of mapping: concept mapping, instance mapping, relation mapping, property mapping.

A. Concept mapping

To construct table in DB, we must describe the structure of table which composed on columns and records. In general [21], a concept in ontology maps to table in DB but in our approach concept mapping takes two cases, first case concept maps to table and the second case concept maps to data in the table:

- Case1: Concerns concept explained using a sub-concept. This type of concept found in non canonical ontology. Also, this case concerns concepts having instances. This type of concepts exists in canonical and non canonical ontology. This two kinds of concepts map to table in DB because it contains the basic elements of table: data can be represented using instance or sub-concept and column can be represented using property (see figure 1).

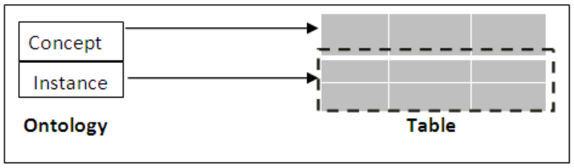


Fig. 1. Case 1: Map concept to table

- Case2: Concerns concept without instance or concept not related to sub-concept. This type can be found in both canonical and non canonical ontology. This case cannot map to table because is not containing element necessary to construct table. It only describes a concept father to which it belongs. This type of concept maps to data in the database. These data belong to the table that represents the concept father (see figure 2).

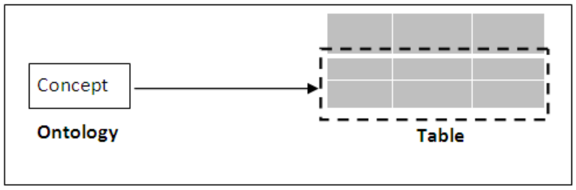


Fig. 2. Case 2: Map concept to instance

B. Property mapping

The concepts properties describe instance and columns in DB table provide the structure according to which the rows are composed. So, we can map properties to columns in the table of DB, which describe records stored in table.

C. Relationships mapping

The concept in ontology are related between them, Also table in DB are related. It describes instance relationships that must exist when a class is instantiated. We have three types of relationships in ontology:

- Relationship between primitive concept in canonical ontology: relates only the primitive concepts of ontology.
- Relationship between defined concept in non canonical ontology: relates two sub-concept which describe the same concept.
- Relationship between primitive concept and defined concept in non canonical ontology: the relationship between concept and its sub-concepts.

The differs relationship between concept in ontology maps to relationships between table in DB.

D. Mapping instance

Each instance of concept is generated as record in DB table. Instances are extracted from ontology to store as record in DB table. We have two cases of mapping instance:

- Case1: Concept contains instances map to table as explained in previous section and instance map to record for this table.
- Case2: Concept without instances map to record in table which represents concept father of this concept.

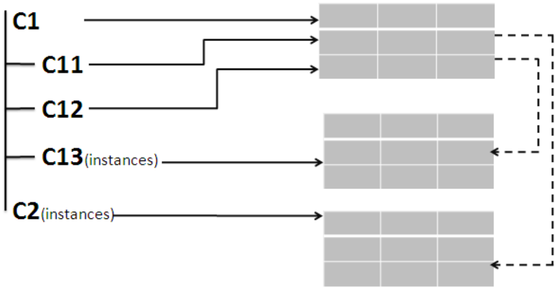


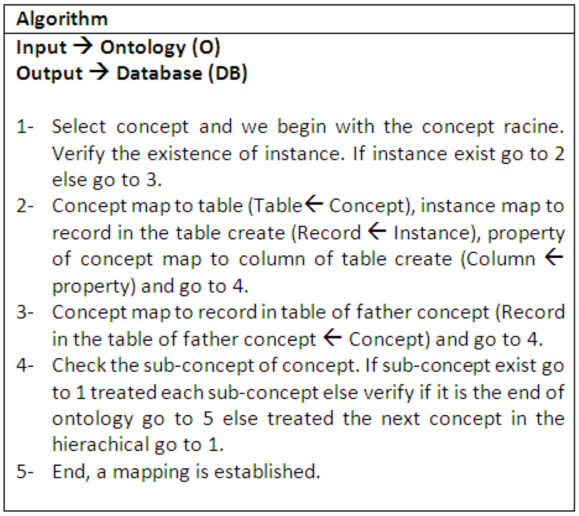
Fig. 3. Example of mapping

The figure (fig 3) explains the mapping components. We illustrate a structure of ontology composed on two primitives concepts (C1, C2) and three sub-concepts (C11, C12, C13). C13 and C2 contain instances. Based on mapping OntoMapDB approach, mapping will be as follows:

- C1: concept related to three sub-concepts. This condition belongs to case 1 of mapping concepts. So C1 maps to table in DB.
- C2: concept contains instance and not related to sub-concept. This condition belongs to case 1 of mapping concept. So, C2 maps to table and instance stored in C2 maps to record in this table.
- C11: it is a sub-concept of C1, it doesn't contain instance and not related to sub-concept. It is the table of concept father which is C1.
- C12: it is the same case as C11.
- C13: it is a sub-concept of C1. It contains instances and not related to sub-concepts. Case1 of mapping concept. So, C13 maps to table and instances map to records in this table.

V. Process mapping

The mapping process is done progressively as follows. It starts by mapping the tables for concepts, columns for properties, instances for records, and then mapping the relationships. Mapping begins with the root of ontology hierarchy and treats each concept and sub-concept. For each concept we verify the existence of instances and the sub-concepts. Instance can describe the data in the DB and the sub-concept describes either data or table in DB. Concept with sub-concepts or instances maps to table in DB and instances map to records in the table. Ech sub-concept must be treated one by one, following the same steps as the concepts. We present in the following the algorithm of mapping process (see fig 4):



```
Algorithm
Input → Ontology (O)
Output → Database (DB)

1- Select concept and we begin with the concept racine.
   Verify the existence of instance. If instance exist go to 2
   else go to 3.
2- Concept map to table (Table← Concept), instance map to
   record in the table create (Record ← Instance), property
   of concept map to column of table create (Column ←
   property) and go to 4.
3- Concept map to record in table of father concept (Record
   in the table of father concept ← Concept) and go to 4.
4- Check the sub-concept of concept. If sub-concept exist go
   to 1 treated each sub-concept else verify if it is the end of
   ontology go to 5 else treated the next concept in the
   hierachical go to 1.
5- End, a mapping is established.
```

Fig. 4. Mapping algorithm

VI. Evaluation of OntoMapDB

To evaluate our approach, we choose Cardiac Adverse Drug Events Ontology (Cardiac ADE) and Heart Failure ontology (HF ontology). We illustrate a part of the mapping Cardiac ADE ontology in fig 5. We construct this ontology that groups the cardiac adverse drug events which are errors or complications occurred in cardiac domain and related to drug use [22]. The HF ontology groups term related to heart failure. We illustrate a part of HF ontology in fig 6 and scheme DB of this part in fig 7.

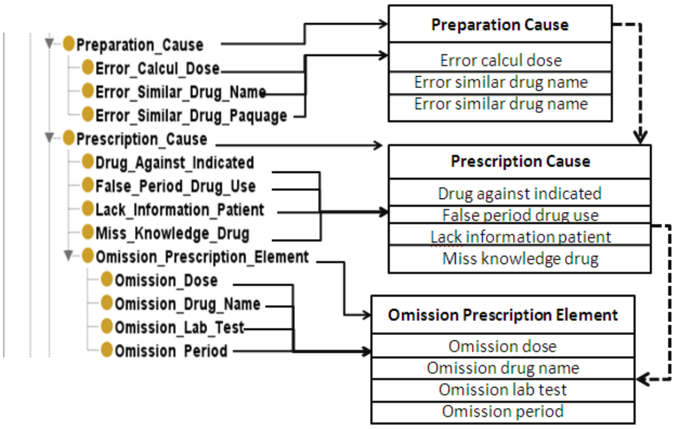
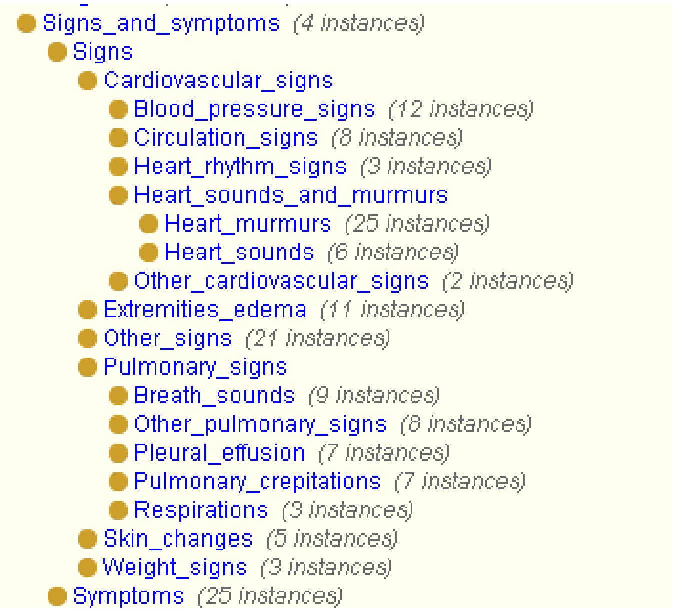


Fig. 5. Example of mapping to ADE ontology

The part of cardiac ADE represented in fig 5 contains 14 concepts, it maps to three DB's table. We note that all concepts does not contain instance. 'Preparation cause' concept contains three sub-concepts and each sub-concept does not contain instance. So, in our approach 'preparation cause' maps to table called 'Preparation cause' table and their sub-concepts map to records in the table. The ’prescription cause’ sub-concept contains five concepts which does not contain instances. ’Prescription cause’ concept maps to table related to ’preparation cause’ table and three sub-concepts map to instance. The sub-concept ’omission prescription element’ contains four sub-concepts. It maps to class related to ’prescription cause’ table and each sub-concept maps to instance in ’omission prescription element’ table. The DB schemas obtained is a proposed schemas which can be used in preventive ADE systems. The developer can modify the schemas of DB resulting. Developer can add properties, delete properties and delete table to proposed schemas.



- ● Signs_and_symptoms (4 instances)
  - ● Signs
    - ● Cardiovascular_signs
      - ● Blood_pressure_signs (12 instances)
      - ● Circulation_signs (8 instances)
      - ● Heart_rhythm_signs (3 instances)
      - ● Heart_sounds_and_murmurs
        - ● Heart_murmurs (25 instances)
        - ● Heart_sounds (6 instances)
      - ● Other_cardiovascular_signs (2 instances)
    - ● Extremities_edema (11 instances)
    - ● Other_signs (21 instances)
    - ● Pulmonary_signs
      - ● Breath_sounds (9 instances)
      - ● Other_pulmonary_signs (8 instances)
      - ● Pleural_effusion (7 instances)
      - ● Pulmonary_crepitations (7 instances)
      - ● Respirations (3 instances)
    - ● Skin_changes (5 instances)
    - ● Weight_signs (3 instances)
  - ● Symptoms (25 instances)

Fig. 6. Part of HF ontology

The part of HF ontology contains 21 concepts. It contains both concepts with instances and others without instances. Concepts contains instances as (signs and symptoms, blood pressure signs, circulations signs, heart murmurs, breath sounds...) map to table in DB, their instances map to records and their properties map to attributes in tables. Concepts without instances as (signs, cardiovascular signs, pulmonary signs, heart sounds and murmurs) map to table in DB, because it related to sub-concepts. The result is illustrated in fig 7.

VII. Conclusion

Knowledge stored in an ontology can be used to assist in generation of DB designs, which called Ontology-Based DataBase (OBDB). The OBDB approaches presented in literature adapted only canonical ontology. In this paper, we present a new OBDB approach called Ontology Map to DataBase (OntoMapDB). It presents a construction rules, used to generate DB’tables, attributes and records. Our approach is adapted to differents types of ontologies canonical and non canonical one, also OntoMapDB is flexible, practical and helpful to Database developer. Our future goal is the development of tool based on this OntoMapDB approach to create OBDB.

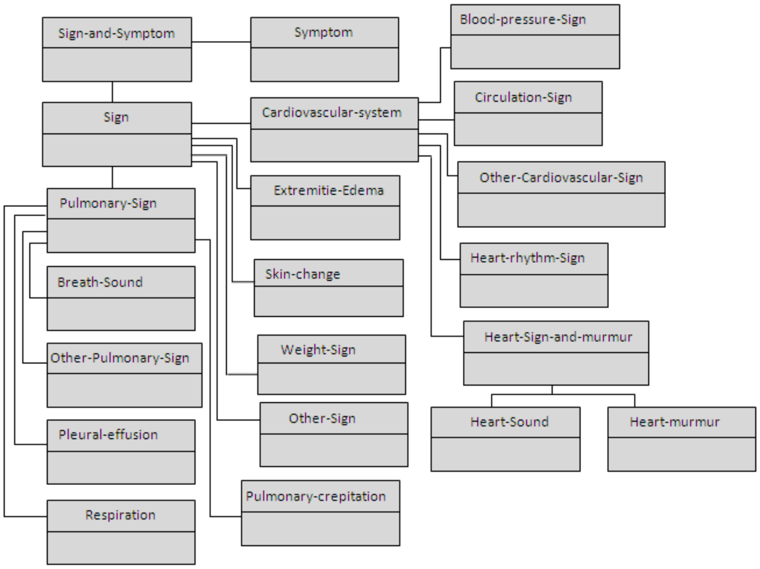


Fig. 7. DB’table scheme of HF ontology part

References

[1] M. Schmidt-Schaub and G. Smolka, “Attributive concept descriptions with complements,” *Artificial Intelligence*, 1991.

[2] T. Gruber, “A translation approach to portable ontologies,” *Knowledge Acquisition*, 1993.

[3] V. Waralak and W. Siricharoen, “Ontology modeling and object modeling in software engineering,” *International Journal of Software Engineering and Its Applications*, vol. 3, pp. 43–60, January 2009.

[4] M. Uschold and M. Gruninger, “Ontologies: Principles, methods and applications,” *Knowledge Engineering Review*, 1996.

[5] G. Pierra, D. Hondjack, Y. A. Ameur, and L. Bellatreche, “Bases de donnes base ontologique. principe et mise en oeuvre,” *Ingeacute;nierie des Systegrave;mes d Information*, vol. 10, no. 2, pp. 91–115, 2005.

[6] H. Dehainsala, G. Pierra, and L. Bellatreche, “L.: Ontodb: An ontology-based database for data intensive applications,” in *In Proc. of the 12th Int. Conf. on Database Systems for Advanced Applications (DASFAA’07). LNCS.* Springer, 2007, pp. 497–508.

[7] G. Studer, B. Richard, and D. Fensel, “Knowledge engінerinng: principales and methods,” *Data Knowl. Eng*, 1998.

[8] D. Bourigault, N. Aussenac-Gilles, and J. Charlet, “Construction de ressources terminologiques ou ontologiques a partir de textes : un cadre unificateur pour trois etudes de cas,” *Revue d’Intelligence Artificielle*, 2004.

[9] M. Uschold and M. King, “Towards a methodology for building ontologies,” *IJCAI*, 1995.

[10] A. Gomez-Prez, M. Fernandez-Lopez, and O. Corcho, “Ontological engineering with examples from the areas of knowledge management, e-commerce and the semantic web,” *Springer*, 2004.

[11] “Ontoql, un langage dexploitation des bases de donnes base ontologique,” Ph.D. dissertation, Ecole Nationale Suprieure de Mcanique et dArotechnique, 2007.

[12] T. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?” *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995. [Online]. Available: http://linkinghub.elsevier.com/retrieve/doi/10.1006/ijhc.1995.1081

[13] “Intgration de bases de donnes htrognes par articulation priori d’ontologies: application aux catalogues de composants industriels,” Ph.D. dissertation, Ecole Nationale Suprieure de Mcanique et dArotechnique, 2002.

[14] S. Jean, Y. At-Ameur, and G. Pierra, “Querying ontology based database using ontoql (an ontology query language),” in *IN PROCEEDINGS OF OTM CONFEDERATED INTERNATIONAL CONFERENCES (ODBASE06)*, 2006, pp. 704–721.

[15] S. Jean, L. Bellatreche, G. Fokou, M. Baron, and S. Khouri, “Ontodbench: Ontology-based database benchmark,” in *28e journes Bases de Donnes Avances (BDA)*, Clermont Ferrand, 2012.

[16] B. McBride, “Jena: Implementing the RDF model and syntax specification,” in *Proc. of the 2. Int. Workshop on the Semantic Web - SemWeb’2001*, ser. CEUR-WS, S. Staab, Ed., vol. 40, Hongkong, China, 2001.

[17] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle, “The ics-forth rdfsuite: Managing voluminous rdf description bases,” 2001.

[18] L. Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu, “Rstar: an rdf storage and query system for enterprise resource management,” in *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*, D. A. Grossman, L. Gravano, C. Zhai, O. Herzog, and D. A. Evans, Eds. ACM, 2004, pp. 484–491.

[19] M.-J. Park, J. Lee, C.-H. Lee, J. Lin, O. Serres, and C.-W. Chung, “An efficient and scalable management of ontology,” in *Proceedings of the 12th international conference on Database systems for advanced applications*, ser. DASFAA’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 975–980. [Online]. Available: http://dl.acm.org/citation.cfm?id=1783823.1783937

[20] Y. Theoharis, V. Christophides, and G. Karvounarakis, “Benchmarking database representations of rdf/s stores,” in *IN ISWC*, 2005, pp. 685–701.

[21] J. Barrasa, scar Corcho, and A. Gmez-prez, “R2o, an extensible and semantically based database-to-ontology mapping language,” in *in In Proceedings of the 2nd Workshop on Semantic Web and Databases(SWDB2004*. Springer, 2004, pp. 1069–1070.

[22] M. Chan, F. Nicklason, and J. H. Vial, “Adverse drug events as a cause of hospital admission in the elderly,” *Intern Med J*, vol. 31, pp. 199–205, May 2001.