2009 First International Conference on Advances in Databases, Knowledge, and Data Applications

Towards Connecting Database Applications to Ontologies

Chimène Fankam, Stéphane Jean, Guy Pierra, Ladjel Bellatreche, Yamine Aït-Ameur
LISI/ENSMA Poitiers University
Futuroscope 86960 France
(fankamc, jean, pierra, bellatreche, yamine)@ensma.fr

Abstract

Most database applications are designed according the ANSI/SPARC architecture. When it is used, a large amount of semantics of data may be lost during the transformation from the conceptual model to a logical model. As a consequence exchanging/integrating various databases or generating user interfaces for data access become difficult. Ontologies seem an interesting solution to solve these problems, since they allow making explicit the semantics of data. In this paper, we propose an ontology-based approach for designing database applications, and then, for representing explicitly the semantics of data within the database. It consists in extending the ANSI/SPARC architecture with the ontological level. Note that this extension may also be added to existing applications designed according to the ANSI/SPARC architecture, since it preserves an upward compatibility.

1 Introduction

Currently, databases are the main tool used for large data storage. Traditionally, designing a database involves four data models: (1) conceptual modeling, (2) logical modeling, (3) physical modeling, and (4) external modeling. A conceptual model (CM) represents knowledge on a given domain. It is characterized by (a) the domain addressed (universe of discourse, UoD), (b) the formalism used to define it (e.g., entity-relationship model), and (c) the modeling point of view that corresponds to specific user requirements and queries. The CM is translated into a logical model corresponding to a data specification implemented in a database system. This translation is obtained by applying transformation rules which produce data descriptions conforming to database theory requirements (normal forms). The physical model defines how data are stored and can be accessed (e.g., index mechanisms). External models or user's views allow a database designer to adapt data according to user's requirements. This database design process follows the traditional ANSI/SPARC architecture proposed by Bachman [1]. This architecture led to several implementations and it has proved its efficiency during the last thirty years. Regarding the semantic exploitation of the resulting data models, this architecture has three major drawbacks:

1. a difficulty related to domain modeling when the designer has no initial knowledge of the targeted domain. For example, in the engineering domain a number of ontologies already exists [15] and could be used to generate CM.
2. a strong dependency of models with designer's style and specific application requirements. Indeed, two conceptual models designed by two different designers targeting the same functionalities are in general, (1) partially different from the point of view of the modeled domain, and (2) largely different from the point of view of the structure of the resulting models. This corresponds to the classical problem of data integration [3].
3. a gap between conceptual and logical models that increases with the discrepancy of the modeling languages. For instance, in relational databases (RDB), the translation from the conceptual model to the logical model involves complex transformation operations. The resulting logical model is very different from the initial conceptual model (e.g., entities and relationships are decomposed into several tables). The semantics of the resulting logical model becomes much less explicit requiring database designer to develop specific application for end-users.

We present in this paper, a new design methodology whose goal is both to help designer in her/his domain modeling task, in particular when domain ontologies already exist and to easy data accessibility and integrability. The basic idea is to enrich the ANSI/SPARC architecture by one more level representing domain ontology. In this approach, the design of a conceptual model (CM) is preceded by the selection and possibly the extension/specialization of existing domain ontology. Then the CM is extracted by specialization from the ontology and it maintains references to that ontology. Note that since every model developed during the design process is linked to the ontology, the meaning of each data is explicitly represented in a database. We call

978-0-7695-3550-0/09 $25.00 © 2009 IEEE
DOI 10.1109/DBKDA.2009.22

such a database an *ontology-based database* (OBDB) and we discuss some aspect of its implementation.

This paper is composed of 7 sections. Section 2 presents a comparison between ontologies and conceptual models. Section 3 proposes our extension of the ANSI/SPARC architecture with the ontological level. The new capabilities of this extension are presented in section 4. Section 5 discusses some implementation of these capabilities. Section 6 presents the state of art. Finally, section 7 summarizes the main results and suggests future work.

2 Ontologies and Databases

Our approach suggests extracting CM from ontologies, it is then necessary to compare domain ontologies and CM. We summarize this comparison in this section.

2.1 Some aspects of Ontology and CM

Several definitions have being proposed for ontology [9, 10]. In our work [13], we will define a domain ontology as *a domain conceptualization in terms of classes and properties that is formal, consensual and referencable*. This definition emphasizes the three criteria that distinguish ontologies from all other models used in Computer Science. An ontology is:
1. *formal* : its contains logical axioms and may be processed by computers; so checking consistency and performing automatic reasoning are made possible
2. *consensual* in a community i.e., several members have agreed upon the concepts represented in the ontology,
3. each concept in the ontology (i.e., class and property) has the *capability to be referenced* through a universally unique identifier. This mechanism allows defining the semantic of a piece of data, whatever are the modeling schema of the ontology and the data model.

When we compare more precisely ontologies with conceptual models, two more differences appear [17, 7].
1. *canonicity of information modeling*. When a domain model is design, various modeling constructs may be used for representing the same information. For instance, a woman may be represented either as an instance of *Person* whose *gender* is female, or as an instance of *Woman*. A crucial characteristic of databases is that all similar pieces of information, (i.e., that belong to the same class) shall be modeled the same way. In the above example, no representation is always better to be represented. The CM define a *canonic language* in the sense that there exists only one description for each information. Contrary wise, most ontology models allow to define conceptual equivalences between different constructs and to use them simultaneously in the same ontology. Fortunately, all ontology models also distinguish the set of concepts that cannot be defined over other concepts. These concepts, called *primitive* concepts, define a canonical language. The other concepts, called *defined* concepts [9] may be defined over the primitives concepts. 2. *the modeling objectives*. CMs *prescribe* information to be represented in a given information system, while ontologies *describe* concepts of a domain independently of any application or information system where the ontology could be used; thus several systems may share the same ontology even if their data model are different, and instances of similar classes may be described differently. It eases ontology usage for data exchange or integration. This last difference is important for our purpose that we further discuss in the next section.

2.2 Classification of Domain Ontologies: the Onion Model

The differences between canonical and non canonical ontologies led to a classification of the various ontologies modeling languages and ontologies. They may be classified in the three following categories.
- *Conceptual Canonical Ontologies (CCOs)* provide concept definitions without redundancy. In CCOs (for example [11]), each information is represented only in a single manner. Note that each conceptual ontology includes a CCO: namely the set of its primitive concepts.
- *Non Conceptual Canonical Ontologies (NCCOs)* contain not only primitive concepts (canonical) but also defined concepts, i.e. concepts for which the ontology provides a complete axiomatic definition by means of necessary and sufficient conditions expressed in terms of other - primitive or defined - concepts. NCCOs are extension of CCOs.
- *Linguistic Ontologies (LOs)* define terms appearing in the universe of discourse of a given domain. In addition to these textual definitions, a number of linguistic relationships (synonym, hyponym, etc.) are used to capture in a semi-formal way relationships between terms. An example of LO is Wordnet [1].

These three categories of ontologies can be combined into a layered model, called the Onion Model and shown in Figure 1. At the heart of this model is a CCO. It provides a formal basis to model and to exchange efficiently the knowledge of a domain. From the primitive concepts of a CCO, a NCCO may be designed. This NCCO provides constructors relating different conceptualizations. Finally, a natural language representation of NCCO concepts, possibly in the various natural languages where these concepts are meaningful, can be provided by a LO.

From a database point of view, each category of ontology offers particular capabilities:

[1] http://wordnet.princeton.edu/

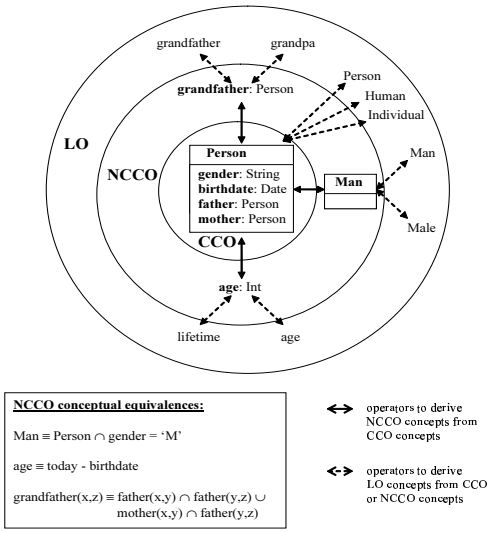

Figure 1. Example of the Onion Model of domain ontology

- CCOs provide a canonical and precise description of all the concepts of a given domain. They provide in particular a formal basis for data exchange between different data sources; every subset of a CCO fulfill the canonicity requirements of a CM.
- Operators of NCCO allow in particular, to connect several ontologies defined over the same domain: non-canonical concept being defined over canonical concept, each non-canonical concept may be represented as a view over a database that represents the relevant canonical concepts.
- LOs provide linguistic capabilities over both categories of concepts (primitive and defined) of the domain. From a database point of view, a LO would provide natural language access to the database content.

Example 1. Figure 1 presents an ontology composed of the three layers of the Onion Model. The CCO part is composed of the class `Person` with its properties `gender`, `birthdate`, `father` and `mother`. From this class and these properties, the non canonical concepts `Man`, `Woman`, `age` and `grandfather` are built. The conceptual equivalences, used to define them and expressed by logical expressions, are given in the bottom part of this figure. Finally, the LO part of this ontology is composed of the various names in English that can be used to refer to the defined concepts. For instance, the `Person`, `Individual` or `Human` are synonymous names for the class `Person`. At the LO level also appear some words that can hardly be formally described since their meaning depend upon the locutor. These words include e.g., boy, child or miss.

3 Applying Ontologies to Databases: a Proposed Extension of the ANSI/SPARC Architecture

The major objectives of a database are to ensure an efficient management of data and to provide access to data independently of their physical representation. The ANSI/SPARC architecture has been proposed to fulfil these objectives. It distinguishes two main access levels:
- the physical level defining how data are stored and managed using a file management system;
- the logical level defining how data are structured on the database data model (e.g., relational or object model).

When designing a database according to this architecture, a large amount of data semantics may be lost during the transformation from the conceptual model to a logical model. Moreover, the meaning of the CM is not formally documented, and thus it cannot be stored in the database. To solve these difficulties, reference to an ontology appears as a relevant solution. We then propose to extend this architecture (see Figure 2) with the *ontological level*. This level defines explicitly data semantics. Moreover, the CM will be modeled by referring their absolute identifier to the ontology concept to which they correspond.

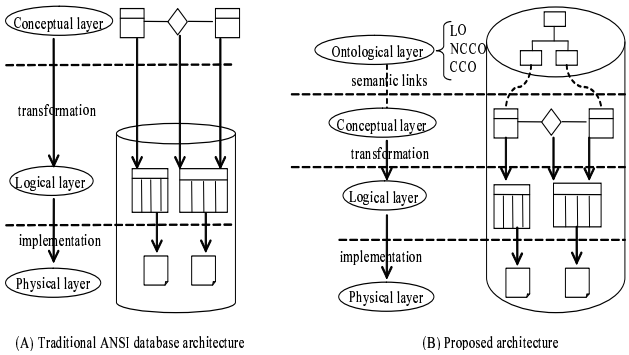

Figure 2. Proposed extension of ANSI/SPARC architecture

Figure 2(A) presents the traditional database architecture. A conceptual model is represented in a modeling language like Entity-Relationship. Then, it is often used to generate automatically the logical model of data. This logical model is represented at the physical level by a set of files. In Figure 2(B) we propose to extend this architecture with the following elements.
- **Ontological level**. It is composed of one (or several if the scope of the system encompasses the domain of several existing ontologies) ontology defining the concepts of the UoD in terms of well-defined classes and properties. It is

possible that this ontology needs to be specialized to represent some concepts w.r.t the system requirements. In this case, our method consists in extending the pre-existing ontology to develop a local ontology that covers all the requirement of the target application. This ensures that all concepts extracted to define the CM will have, at the ontological level, the same formal and precise definition.

- **Subsumption links**. They link the ontological and conceptual levels, thus defining the set of ontology concepts used to fulfill applications requirements. The meaning of these links, represented in the CM by the absolute identifier of an ontology concept, is that the CM concept is equal or is a special case of the referenced ontology concept (i.e., subsumption). Notice that non-canonical concepts are not referenced for defining the CM. A CM being necessary canonic, extracting or specializing only from the canonic part of the ontology will ensure that the result will be canonic. Non canonic concepts will be referenced, in a second step, for defining the meaning of views.

- **Logical level**. As usual; the logical level is generated from the conceptual level by a set of rules. If these rules leads to several tables for the same class, a view is automatically created that represents (virtually) the class instances.

Example 2. Figure 3 illustrates the proposed extension of ANSI/SPARC architecture. The ontological level is composed of the classes `Person` (canonical) , `Man` and `Woman` (non canonical) of the previous ontology (see example 1). The subsumption links define the fact that a `customer` in the CM is a `person` in the meaning of the ontology. Moreover, a `customer` has a subset of the `person` properties (`gender` and `father`). Then, as usual, this class is transformed into tables at the logical level and its properties are translated to columns of this table. One single table being generated, no view needs to be created.

The four levels of the proposed architecture (physical, logical, conceptual and ontological) are intended to be all implemented in the database. Our approach offers specific capabilities that we discuss in the next section.

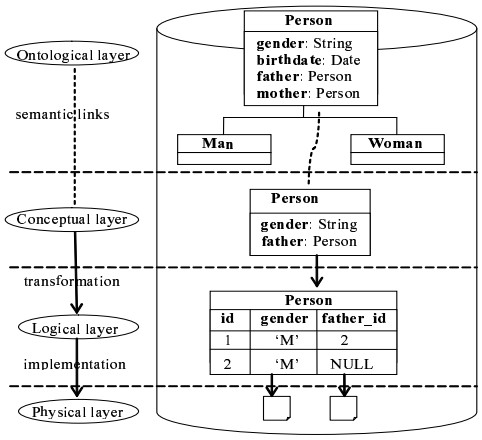

Figure 3. Example of the proposed extension of ANSI/SPARC architecture

4 New Capabilities of the Proposed Database Architecture

Compared to the ANSI/SPARC architecture, the proposed database architecture that we have implemented in our prototype *OntoDB2* offers five major new capabilities.

4.1 Capabilities resulting from the four level architecture

One of the most important characteristics of the ANSI/SPARC architecture is to separate the physical and logical representation of data. Our proposed architecture adds one more independency. Representing both the conceptual and the ontological description of the represented data, our new levels make it possible to exploit data independently of their logical schema. Provide that an ontology query language is used. In our prototype we use OntoQL [12].

Capability 1: The database management system (DBMS) allows expressing query at the ontological level independently of the logical representation (schema) of data.

Another fundamental characteristic of the ANSI/SPARC architecture is to define an external level. This level defines external schemas (views) that reflect user's perception of the application domain data (e.g., woman in place of person [gender = female]).

In our proposed architecture, the NCCO layer of ontologies allows each user to represent its own perception of the application domain by using non canonical concepts. Thus, defining and exploiting such concepts is a new capability of this architecture.

Capability 2: The DBMS supports the definition of non canonical concepts defined as views over the canonical concepts of ontology. Queries may be expressed using canonical and non canonical concepts and the query engine interprets them in terms of canonical concepts.

The last layer of the Onion Model is composed of the LO part. The LO part associates to each concept one or several terms and textual definitions. These linguistic definitions (often given in different natural languages) allow human users to understand ontology and names reference ontology concepts. To make it easy for members of different countries to use the same ontologies, the database exploitation language may support the definition of multilingual LO. It is the case in the OntoQL language that we use.

Capability 3: The DBMS may support the definition and

exploitation of linguistic definitions of concepts that may be defined in different natural languages.
The above capabilities result from the three layer decomposition of the ontological level according to the three layers of the Onion Model. Another characteristic of our proposed architecture is to enforce upward compatibility with the ANSI/SPARC architecture. This feature leads to other capabilities.

4.2 Capabilities for Preserving Compatibility with the ANSI/SPARC Architecture

Our proposed architecture extends the ANSI/SPARC architecture, and thus, it also includes the usual logical level. Since our objective is to define a DBMS for manipulating data at the different levels of our proposed architecture, our system supports manipulation of data not only at the ontological level but also at the logical level.
Capability 4: The DBMS permits the manipulation of data also at the logical level preserving SQL compatibility.
Designing a layered architecture has the drawback to increase the complexity of data processing in upper levels. Thus, to optimize query processing at the ontological level, the DBMS shall provide an access to the lower level, i.e. the conceptual level. It is done in OntoDB2.
Capability 5: The DBMS handles access to data at the conceptual level from the ontological level.
Capabilities 1, 2 and 3 allow expressing ontological queries using canonical, non canonical and linguistic definitions provided by ontology. Capability 4 permits to access the logical level using SQL. Capability 5 allows accessing the conceptual level from the ontological level in order to optimize data processing.

In this section we have presented capabilities resulting from the implementation of the proposed extension of the ANSI/SPARC architecture. These capabilities are in the process to be implemented in the OBDB OntoDB2 whose software architecture is outlined in [8]. In particular, OntoDB2 supports the definition of non canonical concepts (capability 2). We explain in the next section how the non canonical constructs in an ontology may be used for creating theses views.

5 Representation of Non Canonical Concepts

The non canonical class constructions available in the ontological layer define equivalences between non-canonical classes and the canonical class to which they correspond. Thus, they define the semantics of views.
Example 3. In figure 1, the non-canonical concept `Man` is defined as `Person` $\cap$ `gender` = 'M'. In the database, all persons are represented as `customer` with properties `gender` and `father`. Thus if `Man` is requested, the system may generate the view :
CREATE VIEW `Man` AS
SELECT * FROM `customer` WHERE `gender` = 'M'

In fact, according to the ontology model used, various equivalence constructors are defined. These operators make it possible to enrich the external layer with new derived semantic concepts for the data stored in the database.

The main problem when representing non canonical concepts is that constructors provided by ontology models are numerous and various. Constructors of non canonical concepts we have chosen to consider are presented in table 1. According to their origin, these constructors can be grouped in four categories:
- constructors that define classes come from Description Logic. The formal description of these constructors can be exploited to generate the corresponding views.
- logical rules. They come from Frame Logic. To implement them we need recursive query languages. Our prototype being on PostgreSQL, We did not implement this capability.
- algebraic expressions. They come from the data processing community. For instance : `diameter = 2*radius`. This capability has been implemented in our prototype.

We outline below how the mechanisms in Table1 are currently implemented.

Constructor	Ontology Model
Constructors of classes by boolean expressions (Union, Intersection, Complement)	OWL
Constructors of classes as restrictions (of domain, value or cardinality) see example 3	OWL
Characteristic of properties (inverse, symmetric, transitive)	OWL
Logical rules (grandfather (x,y) :- father(x,z) $\wedge$ father(z,y))	F-Logic
Algebraic expressions	PLIB

Table 1. Constructors of non canonical concepts proposed by ontology models

5.1 Defined Classes

To manage defined classes, the goal is to avoid redundancy of data. Defined classes being defined using set operations (union, intersection) or restrictions operators, our approach (see example 3) consists in computing their extension as a view rather than storing them. This approach is

implemented using the view mechanism (with triggers) that is used to compute union, intersection or selection over sets.

5.2 Algebraic Characteristics of Properties

In this case, following the closed world assumption, we materialized all the data.
- Symmetry: we materialize a priori all data. This approach is easy to implement and support update and delete of data.
- Transitivity: materializing all the derived facts is also the solution we adopted. We assume at the beginning, that all data are materialized. Thus, when a new relation concerning a transitive property is added, the new facts are materialized by a trigger in a non recursive manner as follow:

```
... ON INSERT P(x,y)
        FOR EACH EXISTING PAIR P(i,x)
                ADD a new pair P(x,j)
        FOR EACH EXISTING PAIR P(y,j)
                ADD P(x,j)
```

Notice that a difficulty is that we cannot remember why some pair is in the database. But anyway, there are no reason why dropping a particular pair should remove other pairs. It depends upon the new state of the world that is unknown and should be specified by the user.

5.3 Algebraic Expressions

Like for defined classes, the value of a derived property is computed by evaluating its expression. This expression is generated by the system and its evaluation is also encapsulated by a view.

6 Related work on ontologies and databases

Three approaches have been proposed for using together ontologies and databases.

The first approach, that we call *a posteriori*, uses an ontology for accessing to a pre-existing database. In particular, a language called R2O [2] has been proposed for defining the mapping from a relational schema to the ontology. This approach is not intended to be used for designing a new database even if the R2O language may be used for expressing the mapping between ontology and the logical model.

The second approach suggests using an existing ontology for helping a designer to define a conceptual model. We call this approach the *a priori* approach. Two research teams have proposed such an approach. Roldan-Garcia and its colleagues [16] just outline a methodology where an existing ontology is extended to address the application needs, and then a subset is selected to define the CM. But nothing is define about the relationships between the various models, nor on the representation of the ontology in the database. Sugumaran and its colleagues [18] propose a natural language processing approach for extracting from the textual description of the system requirements, and from a linguistic ontology those ontology terms that are suggested in the CM. On the basis of practical experimentation, the paper proves the efficiency of this approach. But nothing in the proposal allows neither to represent formally the link between ontology and CM, nor to represent the ontology in the model. Our approach is a priori approach, but it defines formally the link between ontology and data. Moreover, the database may contain ontology-based data, the ontology and provides access to data through ontology.

The third method results from the growth of RDFS and OWL in the Semantic Web. The data defined in this context are represented as ontology class instances by means of RDF triples. This approach, that we call *ontology-structured database*, directly uses the ontology instance structure for defining the instance model. This model may be either a single triple table both for ontology and for data [4, 6], or a set of ontology-specific tables for ontologies and unary tables for class belonging and binary tables for property values [5, 14]. The major drawbacks of all these approaches from database design perspective is that the data structure is completely defined by the ontology. The database designer has no control at all : once the ontology is selected the database structure is fixed. Moreover, most of these approaches do not require data to be described in a canonic way, thus either reasoning is requested for answering query or data are duplicated. Our goal is also to embed the ontology in the database, but we need the capability first to extend the ontology for missing concepts, then to choose the CM as a subset of this extended ontology. It should be noticed that the proposal from Oracle group [6] permits to manage together the ontology in a particular RDF structure and data in a particular structure chosen by the database administrator. Thus this structure should also be a candidate for storing the result of our proposed method.

7 Conclusion and Future Work

In this article we have presented a new design method that extends the traditional ANSI/SPARC architecture. An additional layer is added on top of the layered ANSI/SPARC architecture: the ontology layer. This layer describes the semantics of the data prescribed in conceptual models independently of any conceptual modeling language and of any database model. The reference mechanism offered by ontologies is used to annotate concepts of the CM. Indeed, each CM concept references, by its unique identifier, the semantic definition available in ontologies. This method is mainly interesting when some domain ontolo-

gies already exist in the application domain. Nevertheless, it is seldom the case that a pre-existing domain ontology already addresses all the application requirements. Thus our method allows extending and/or specializing the existing ontology into a local ontology that covers all the application requirements.

When this additional design level is also implemented in the database, as we did within our OntoDB2 prototype, the resulting database is an ontology-based database (OBDB). When equipped with this level, OBDBs represent the semantics of data and support access to data at the knowledge level. It is in particular possible to implement an ontology query language such as OntoQL, to query data at the ontological level. Notice that processing in such way preserves an upward compatibility with the traditional ANSI/SPARC architecture; logical data being also possibly queried in SQL.

In, this article we have mainly focused on the definition of the ontology layer. Canonical, non canonical and linguistic ontologies can be represented in this layer. Their presence allows a user or a designer to access the database through rich semantics definitions ignoring the logical and physical database models. The more ontologies are rich in non canonical concepts, the more a user can access the database using different semantic point of views. We are currently validating the OntoDB2 prototype that support all the capabilities defined in this paper.

References

[1] C. W. Bachman. Summary of current work - ansi/x3/sparc/study group - database systems. volume 6, pages 16–39, 1974.

[2] J. Barrasa, scar Corcho, and A. Gmez-Prez. R2o, an extensible and semantically based database-to-ontology mapping language. *Second Workshop on Semantic Web and Databases (SWDB'2004).*, August 2004.

[3] L. Bellatreche, G. Pierra, D. Nguyen Xuan, H. Dehainsala, and Y. Ait Ameur. An a priori approach for automatic integration of heterogeneous and autonomous databases. *International Conference on Database and Expert Systems Applications (DEXA'04)*, pages 475–485, September 2004.

[4] B.McBride. Jena: Implementing the rdf model and syntax specification. *Proceedings of the 2nd International Workshop on the Semantic Web*, 2001.

[5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. Hendler, editors, *Proc. of the 1st Int.. Semantic Web Conference (ISWC'02)*, number 2342, pages 54–68. Springer Verlag, July 2002.

[6] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An efficient sql-based rdf querying scheme. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1216–1227. VLDB Endowment, 2005.

[7] H. Dehainsala. *Explicitation de la sémantique dans les bases de donnes : Le modèle OntoDB de bases de données à base ontologique*. PhD thesis, LISI/ENSMA et Université de Poitiers, 2007.

[8] C. Fankam. OntoDB2: Support of Multiple Ontology Models within Ontology. In *Proceedings of the EDBT 2008 PhD Workshop. Co-located with the 11th International Conference on Extending Database Technology (EDBT'08)*, 2008.

[9] T. R. Gruber. *Formal ontology in conceptual analysis and knowledge representation. Chapter: Towards principles for the design of ontologies used for knowledge sharing.* Kluwer Academic Publishers., 1993.

[10] N. Guarino and R. Poli. Formal ontology in conceptual analysis and knowledge representation. *Special issue of the International Journal of Human and Computer Studies*, 43(5/6):625–640, 1995.

[11] IEC61360-4. Standard data element types with associated classification scheme for electric components - Part 4 : IEC reference collection of standard data element types, component classes and terms. Technical report, ISO, 1999.

[12] S. Jean, Y. Aït-Ameur, and G. Pierra. Querying ontology based database using ontoql (an ontology query language). In *Proc. of On the Move to Meaningful Internet Systems : CoopIS, DOA, GADA, and ODBASE, OTM Confederated Int. Conf. (ODBASE)*, pages 704–721. Springer, 2006.

[13] S. Jean, G. Pierra, and Y. Aït-Ameur. *Domain Ontologies : a Database-Oriented Analysis*, volume 1 of *Lecture Notes in Business Information Processing*, pages 238–254. Springer Berlin Heidelberg, 2007.

[14] J. Lu, L. Ma, L. Zhang, J.-S. Brunner, C. Wang, Y. Pan, and Y. Yu. Sor: a practical system for ontology storage, reasoning and search. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 1402–1405. VLDB Endowment, 2007.

[15] G. Pierra. Context representation in domain ontologies and its use for semantic integration of data. *Journal Of Data Semantics (JODS)*, pages 173–210, 2008.

[16] Roldan-Garcia, M. Navas-Delgado, Aldana-Montes, and J. Aldana-Montes. A design methodology for semantic web database-based systems. *Third Int. Conf. on Information Technology and Applications(ICITA'05)*, 1:233–237, 2005.

[17] P. Spyns, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. volume 31, pages 12–17, New York, NY, USA, 2002. ACM Press.

[18] V. Sugumaran and V. C. Storey. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. In *ACM Trans. Database Syst.*, volume 31, pages 1064–1094, New York, NY, USA, 2006. ACM Press.