OntoDB: It is Time to Embed your Domain Ontology in your Database (Demo Paper)

Stéphane Jean, Hondjack Dehainsala, Dung Nguyen Xuan, Guy Pierra, Ladjel Bellatreche, Yamine Aït-Ameur

Proc. of Database Systems for Advanced Applications (DASFAA'2007),
Bangkok, Thailand, April 9-12, 2007.

OntoDB: It is Time to Embed your Domain Ontology in your Database

Stéphane Jean[1], Hondjack Dehainsala[1], Dung Nguyen Xuan[1], Guy Pierra[1], Ladjel Bellatreche[1], Yamine Ait Ameur[1]

LISI/ENSMA - Poitiers University - France - E-mail : familyname@ensma.fr

Abstract. This demonstration presents OntoDB, a prototype that allows to store explicitly in the database not only the data, but also the conceptual model defining the structure of data and the domain ontology representing the meaning of data. The demonstration illustrates three main functionalities of OntoDB: (1) a storage of a domain ontology and database content in the same repository, (2) the possibility of querying databases at ontology level, and (3) an automatic integration of heterogeneous data sources referencing/extending the same domain ontology.

1 Introduction

Traditionally, the process of database application design goes through a chain of three major steps: conceptual, logical and physical. The conceptual model (CM) is the core of the application development. Its basic constructs (entity, relationship between entities) are associated with semantics which can be understood intuitively by designers and users. This model is then translated into a logical model. Once this translation done, CM is usually discarded from the design chain. Consequently, application semantics described by this CM may be lost. Other work on ontology was undertaken in knowledge modeling [4]. Contrary to CM, an ontology aims to *describe* in a consensual way the whole knowledge of a domain. This description is agreed and shared by domain experts allowing them to understand each other. When such an ontology exists, the process of database design no longer needs to create completely new conceptualization, but it just *needs to extract or to specialize from the domain ontology pieces of information that are relevant for the application to be designed*. We call this approach *ontology-based modeling*. Recently, Sugumaran et al. work [7] shows how domain ontology can be used to assist in the generation of complete and consistent database conceptual design. Several approaches and systems were proposed to store in the same database, data and the ontologies describing their meanings [1, 2]. In this demonstration, we present one of these systems named OntoDB [3]. By storing the conceptual model defining the structure of data, OntoDB is the only one to follow the ontology-based modeling approach. Moreover, we have shown in [3] that it outperforms other systems for a set of queries.

2 OntoDB Components

OntoDB represents explicitly: (1) ontologies, (2) data structures, (3) data, (4) the links between the data and their schema and (5) the link between schema

This work has been partly supported by the French ANR under grant ANR05RNTL02706 (e-Wok-Hub).

and the ontology. Before defining the architecture of our prototype, we present the three objectives assigned to our architecture model: (1) it shall support an automatic integration and management of heterogeneous populations whose data, schemas and ontologies are loaded dynamically, (2) it shall support evolutions of the used ontologies (adding new classes, new properties, etc.) and of their population schemas, and (3) it shall offer data access, at the ontology level, whatever the type of the used DataBase Management System (DBMS) (relational, object-relational or object). Taking in account these objectives, our architecture is composed in four parts, where part 1 (meta base or system catalog) and part 2 (content) are traditional parts available in all DBMSs, and part 3 (ontology) and part 4 (meta schema) are specific to OntoDB (figure 1).

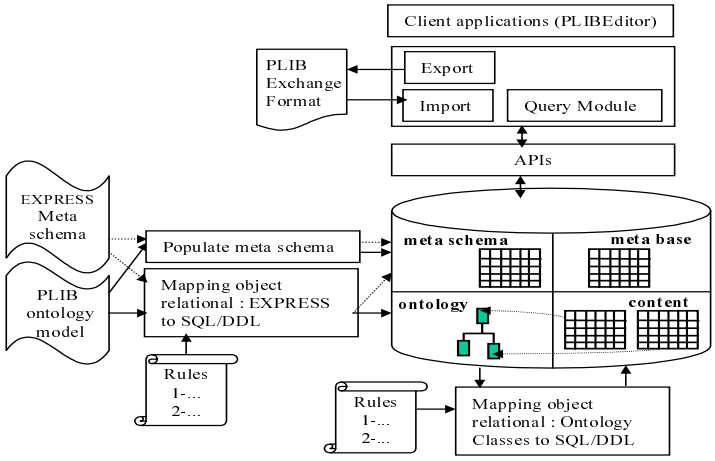

Fig. 1. System Architecture

Ontology part allows to represent ontologies in the database. When ontology model is object oriented and the target DBMS is relational, its logical schema is defined using an object/relational mapping. The *meta schema part* records the ontology model into a reflexive meta model. For the ontology part, the meta schema part plays the same role as is played by the *meta base* in traditional DBMSs. Indeed, this part allows: (1) a generic access to the ontology part, (2) support of evolution of the used ontology model, and (3) storage of different ontology models (OWL, PLIB [6], etc.). The link between ontology, meta base, and content parts is established using a global universal identifier mechanism associated to classes and properties of ontologies.

3 System Implementation

This section shows the implementation of each part of OntoDB. APIs to access OntoDB and modules of client applications are also described (see figure 1).

1. OntoDB. OntoDB is implemented on PostgreSQL7.4 and the (multilingual) PLIB ontology model (POM) is specified in EXPRESS (a formal OO modeling language associated with an environment similar to Meta Object Facility). To implement ontology part, POM has been mapped to a logical schema by a program generator. It is based on defined transformation rules between EXPRESS concepts and SQL/DDL. The logical schema of the meta schema part is also generated automatically by re-using an object relational generator. Concretely, the generator receives as input parameter an EXPRESS meta model of EXPRESS and returns a set of tables representing the meta model. Then the meta schema part is populated with the POM and with itself as data. To define the content part logical schema, the database designer selects a subset of the ontology that represents its CM and then the logical schema is generated by another object relational mapping which takes the CM as input parameter.

2. Import module. Like OWL in XML, the POM allows to represent and exchange both ontologies and ontologies individuals (instance data) in EXPRESS exchange format. The *import module* allows to read a population of the POM (ontology + data) and to store it in the database. If the ontology already exists in the database, it can be potentially updated (version management) and its instances will be automatically integrated in the existing population of ontology instances.

3. Export module. It is dual of the import module. It allows to extract a subset of an ontology from the classes of the ontology in the database, with or without the associated content. This module combined with import module allows to automatically *migrate* instances of a database to another.

4. Ontology and content edition module. It allows to dynamically display and create classes and their properties in the ontology and to dynamically create and visualize objects of these classes. Since each data is associated to an ontological element which defines its meaning, it becomes possible to access the data through their meaning. The interface offered by this module, called *PLIBEditor* is appropriate to any ontology and any population of classes.

5. APIs. Almost all the modules of figure 1 access OntoDB using a three layered API that we have implemented. The first layer is a generic API defined at the meta level (all parameters are strings). It allows to create instances independently of any model in any part of the database, but without any user control. The second layer specific to the POM, is composed of java classes generated automatically, one for each POM entity. This API calls functions of the first API. The last layer is also generated automatically to represent ontology classes as java instances. Note that most of the developed programs, modules and API are not POM-specific but they may be specialized for any ontology model after its description in the EXPRESS language.

The OntoDB architecture requires new query functionalities more those offered by traditional languages like SQL2003. We developed a query language, called OntoQL that allows to query data in terms of concepts of the ontology using its expressive power (multilingual, polymorphism, etc.) [5]. Moreover, it provides a way to extract a part of the ontology with its associated content.

4 The Scenario to be Demonstrated

Our demonstration is based on an shared ontology (SO) defining concepts of the LMD (License, Master, Doctorate) university course. The LMD system has been established in order to harmonize diplomas in the *European Union*. The scenario of our demonstration follows 5 mains steps: **(1) Description of the used shared ontology**: the process of editing ontologies and the characteristics of the POM are shown as well. Our ontology editor (PLIBEditor) is used as client application. **(2) Definition of local ontologies from SO** : two different databases (representing two universities) specialize the SO to define the particular concepts existing in their own course. **(3) Extraction of conceptual models derived from the ontology**: the conceptual models of the two different databases are designed from their local ontologies. Several instances of students are described (using the ontology concepts) and inserted in both databases. **(4) Automatic integration of information**: shows the process of students data integration of the two universities. **(5) Query processing**: a set of retrieval queries using our QBE interface is executed on the integrated data. We show specific queries expressed in different natural languages on data involving ontology concepts that use the expressive power of the ontology model (like inheritance, composition, ...).

Additionally, in our demonstration, we present PLIBEditor that allows to manage (create, delete, import, export, query, etc.) ontologies and ontology-based data stored in OntoDB. The use of the OntoDB architecture to conceptually design a database using domain ontologies is also demonstrated. For more details, refer to our Web site http://www.plib.ensma.fr/plib/demos/ontodb/, some "flash" demonstrations and snapshots are presented.

References

1. Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, and Karsten Tolle. The ics-forth rdfsuite: Managing voluminous rdf description bases. In *2nd International Workshop on the Semantic Web (SemWeb'01)*, 2001.
2. Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference (ISWC'02)*, pages 54–68, July 2002.
3. Hondjack Dehainsala, Guy Pierra, and Ladjel Bellatreche. Ontodb: An ontology-based database for data intensive applications. In *Proceedings of Database Systems for Advanced Applications, 12th International Conference (DASFAA'07) (to appear)*, 2007.
4. Thomas Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 7, 1993.
5. Stéphane Jean, Yamine Aït-Ameur, and Guy Pierra. Querying ontology based database using ontoql (an ontology query language). In *Proceedings of OTM Confederated International Conferences (ODBASE'06)*, pages 704–721, 2006.
6. Guy Pierra. Context-explication in conceptual ontologies : The PLIB approach. In *Proceedings of Concurrent Engineering (CE'03)*, pages 243–254, July 2003.
7. Vijayan Sugumaran and Veda C. Storey. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems (TODS)*, 31(3):1064–1094, September 2006.