Creating and Managing Domain Ontologies for Database Design

Vijayan Sugumaran

Department of Decision and Information Sciences
School of Business Administration
Oakland University
Rochester, MI 48089
Phone: (248) 370-2831
Email: sugumara@oakland.edu

Veda C. Storey

Computers and Information Systems
Georgia State University
Box 4015
Atlanta, Georgia 30302
Phone: 404-651-3894
Email: vstorey@gsu.edu

Abstract: Although ontologies have been proposed as an important and natural means of representing real world knowledge for the development of database designs, most ontology creation is carried out on a manual basis. To be truly useful, a repository of ontologies, organized by application domain is needed, along with methodologies for integrating the use of the ontologies into database design methodologies. This research proposes a methodology for automating the creation and management of ontologies. An architecture for an ontology management system is proposed.

1 Introduction

Database management systems are central to any business because they capture and store the information required to support business operations, whether traditional or e-commerce applications. In fact, databases are considered to be "at the core of any e-commerce web sites." Good designs are important for the efficient operation of a database and are especially important when it becomes necessary to merge two or more databases (traditional or web-based). Database design, however, is difficult and always been considered an art, with the quality of the design highly dependent upon the skill and expertise level of the designer. Good designers are scarce.

A number of tools have been developed that attempt to automate the database design process. These systems, however, do not contain any information about the real world and how it operates. As a result, they rely on the user to provide all of the information about a given application domain (e.g., airline scheduling, hotel reservations, housing rentals). They often ask irrelevant, or seemingly meaningless, questions to the user, making them appear less intelligent than they should. It is believed that the inclusion of knowledge, organized by application domain is the next step in making such design systems "more intelligent" [L197].

The objectives of this research, therefore, are to:

1. Demonstrate why ontologies are important and can be very useful for database design
2. Propose a heuristic-based methodology for creating domain ontologies
3. Propose an architecture for a system to facilitate the use of domain ontologies during the conceptual design process. The system would be used for creating, evolving and managing ontologies, and providing assistance in applying ontologies to database design problems.

The contribution of this research is to provide a methodology for creating ontologies for use in database design. This makes a much richer modeling approach in that more of the semantics and constraints of an application domain are captured. The result is a database that is an accurate representation of the real world created with less designer effort. It will also allow ontologies to be used, evolved, and reused.

2 Related Research

The importance of capturing and representing real world knowledge in information systems has long been recognized in artificial intelligence, software reuse, and database management. Ontologies have been proposed as an important and natural means of representing real world knowledge for the development of database designs. In fact, it is believed that the use of ontologies will fundamentally change the way in which systems are constructed and that designers will have libraries of ontologies from which they can choose appropriate ones [Sw99]. An ontology "defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations between terms" [Ne91]. An ontology may have very high-level terms or be domain-specific [Sw99].

The use of ontologies is found in many areas, including database design [Em99], [KM99]. Most ontology creation, however, is carried out on a manual basis. To be truly useful, a repository of ontologies, organized by application domains is needed, along with methodologies for integrating and using ontologies in the database design process. Most of the current tools support only certain aspects of ontology creation and manipulation, and are application specific. For example, the RiboWeb [Al99] system incorporates ontology constructs related to storing molecular information. Similarly, [Fe99] presents an ontology development environment that is suited for capturing knowledge within the chemical domain. Staab et al. [St01] present an ontology based knowledge management system based on OntoEdit, that primarily focuses on supporting various knowledge processes. This paper presents a heuristics based ontology creation methodology that can be applied to any application domain, in particular, datatabase design.

Although a number of ontologies have been developed, there is no well-accepted methodology for developing ontologies. An analogy can be made to conceptual modeling using the entity-relationship model. The ER model is well-accepted as an effective tool for representing user requirements at the conceptual design level. Even though the constructs of the ER model are well defined and some rules exist for how to apply them, effective use of the model still requires designer expertise or, at minimum, the application of a set of heuristics that have been developed either by experience or through consultation with design experts (e.g., relationship attributes exist only for many-to-many relationships, SSN is usually an attribute as opposed to an entity). Furthermore, some of the rules and heuristics are domain dependent. For this research, we first adopt the definition of an ontology as defining the basic terms and relationships comprising the vocabulary of a topic area as well as the rules for combining them [Ne91]. This research explicitly considers the rules that capture business rules or processes, which are captured through a variety of constraints and meta-level rules. Table 1 shows a partial ontology for the travel domain.

Term	Synonym	Description	Business Rule	Related to
Trip	Travel Product	Type of travel desired	Instances are cruise, flight, train trip, bus tour	Tour
Tour		Travel to and from destination, sight-seeing of points of interest		Traveler, itinerary
Customer		Person interested in trip	Customer who books trip is a traveler	Passenger, tour
Traveler	Passenger	Person who purchases trip	Down payment required before itinerary created	Itinerary, Club, Destination, Customer
Agency		Booking organization		Traveler, tour
Party		People customer travels with		traveler
Itinerary	Schedule	Schedule of events	Departure occurs before arrival	Traveler, tour, party
Ticket		Issued after payment	Traveler must have ticket before departure	Passenger, trip, tour

Table 1. Partial Travel Domain Ontology

Assume a designer is designing a database application in the online travel domain. By visiting travel sites and browsing relevant documentation, the designer might identify concepts such as those shown in the entity-relationship (E-R) model in Figure 1a. However, this model may not be complete or even relevant. The designer consults the ontology to discover that, in this application, there are customers (prospects) and travelers (committed to a trip), as well as other, specific relationships. An important concept is that of an itinerary. Constraints also need to be integrated. The model can be refined to the more complete and accurate description of the application as shown in Figure 1b.

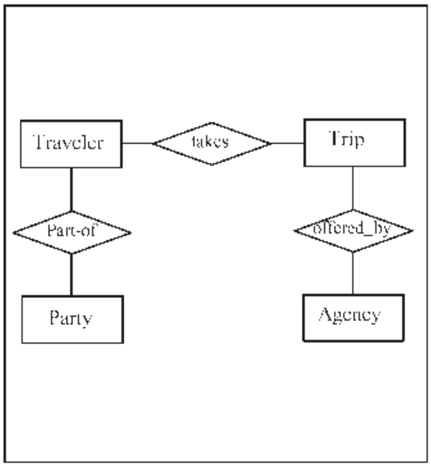

Figure 1a. Initial E-R Diagram

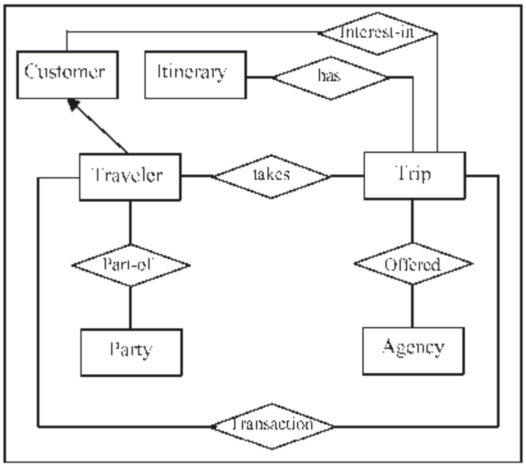

Figure 1b. Revised E-R Diagram

The two tasks in this research are the development of: 1) a heuristic-based methodology for creating ontologies; and 2) an architecture to implement the methodology in an ontology management system that supports the automated creation and evolution of domain ontologies for database design automation.

3 Ontology Creation and Management

This section first presents a methodology for the creation of domain ontologies. It then proposes an architecture for the management of ontologies for use in database design automation.

3.1 Heuristics-Based Ontology Creation Methodology

The methodology for the creation of a domain ontology, as given below, consists of a set of steps, with corresponding heuristics.

Step 1: *Identification of basic terms*
The basic terms of an application domain are identified and definitions given for them and their properties. These are translated into entities in the conceptual model. For example, in the travel domain, obvious terms include traveler, trip, ticket, and itinerary. The properties of a trip include origin and destination. The challenge here is one of completeness, which is accomplished through heuristic 1.1 and by enabling the ontologies to evolve.

Heuristic 1.1 (identification of most frequent terms): Create "use cases" to understand the application domain and the business processes from which relevant terms can be deduced. Use cases are textual narrative descriptions of concepts and processes that exist an application domain and are employed widely in systems analysis [Ja92]. By analyzing existing use cases or creating new use cases in that domain, the designer can identify the important, basic terms.

Heuristic 1.2 (identification of synonyms or closely-related terms): These are generated manually or by an online thesaurus.

Step 2: *Identification of three types of relationships*
An ontology is intended to provide information about an application domain that might not be familiar to a designer. Therefore, the relationships that are identified are more than simple associations between terms. The following heuristics identify the three important types of relationships.

Heuristic 2.1 (relationship between basic concepts): Identification of subset/superset relationships (e.g., traveler is-a customer).

Heuristic 2.2 (relationship between constraints): Identification of relationships between constraints. For example, if two trips overlap, the same traveler cannot be booked on both, therefore, a traveler cannot be a flyer and a cruise ship passenger at the same time.

Heuristics 2.3 (relationship between ontologies): Ensuring that terms have consistent relationships between an ontology and a sub-ontology, in order to allow ontologies to evolve.

Step 3: *Identification of basic constraints*
Constraints capture business rules and semantics associated with one or more terms (e.g. CruiseTraveler (x) $\rightarrow$ prepaid (x)) and relationships. The constraints are of four types: pre-requisite, temporal, mutually inclusive, and mutually exclusive. From these, meta-level constraints can be derived. These constraints are captured in the following heuristics.

Heuristic 3.1 (pre-requisite constraint): One term/relationship depends upon another; e.g., credit card is pre-requisite for paying. Specifically, consider three terms A, B, and C. If term A requires B, and C, then, we define the following pre-requisite constraint:
A $\rightarrow$ B, C. Note that term B may occur by itself, however, for term A to occur, terms B, and C are also needed.

Heuristic 3.2 (temporal constraint): One term/relationship must occur before another; e.g., reservation precedes tour. In the case of terms A, B, and C, if B must precede C, we define the temporal constraint as follows: B $\mapsto$ C.

Heuristic 3.3 (mutually inclusive constraint): One term/relationship requires another for its existence; e.g., to travel to a foreign country a VISA is required, based upon citizenship. In case of the terms A, B, and C, if A requires B and B also requires A, then we have a mutually inclusive relationship between A and B, i.e., these two terms have to occur simultaneously. Then, we define the mutually inclusive constraint as: A $\longleftrightarrow$ B

Heuristic 3.4 (mutual exclusive constraint): One term/relationship cannot occur at the same time as another; e.g., a cruise cannot be listed as being sold out and have availability at the same time. This operates on the constraint for availability and the constraint for sold out. If the terms A and B are mutually exclusive, then only one of them can occur in a given context. We define the mutually exclusive constraint as: A $\not\leftrightarrow$ B

Heuristic 3.5 (meta-level constraints): These are constraints on constraints; e.g., if A pre-requisite for B, then B cannot occur before A under a temporal constraint. (If advertising a cruise is a pre-requisite for selling cruise seats, then, seats cannot be sold before they are available.)

Step 4: *Identification of higher-level constraints capturing domain knowledge*
These are domain-dependent and capture the fact that an ontology is domain-specific. They capture many of the business rules that exist in a given domain.

Heuristics 4.1 (Identify Domain constraint): These are constraints on the domain terms. For example, "double occupancy" applies to a cruise ship reservation, but not an airline flight.

Heuristics 4.2 (Identify Domain dependencies): These are constraints on multiple terms or relationships within an application. For example, an agent cannot book cruises and book airline flights that do not have compatible dates where compatible dates are defined relative to the beginning and ending of a trip (temporal constraints).

3.2 Representation

An entity-relationship formalism can be used to represent ontologies at a conceptual level with the translation from an item and its properties to an entity and its attributes being quite natural. The various types of relationships will appear as relationships among entities, semantic integrity constraints and abstract data types. The internal representation is an UML model. This refers to how one physically stores an ontology.

3.3 Ontology Evolution

Most ontologies are static [PB99]. However, domains in the real world evolve over time (e.g., e-trading is a new concept in the financial domain). Therefore, ontologies should be able to evolve through the addition, deletion, or modification of terms, properties, relationships, and constraints. In addition, the ontologies themselves might need to be subdivided or combined to reflect changing environments. For example, a financial ontology might be split into ones for e-trading, bonds, or stocks. Some of these would have overlapping terms. The constraints need to be checked to verify that none are in violation, and the heuristics checked for consistency.

4 Ontology Management Tool

The architecture of the ontology management system, as shown in Figure 2, consists of two parts, the client side and java-enabled server side. The client is a basic web browser using which the user can browse or modify existing ontologies, as well as create new ones, which are stored in an ontology repository. The server side has four major modules: a) the interface module, b) the browse module, c) the logical module, and d) the physical module.

a) Interface Module: This module is responsible for gathering the input from the user, passing it to the browse or logical module, and sending the results back to the user. It has several sub-modules implemented as java servlets. "Gather browse input" gathers the user's selections during browsing and sends it to the "browse query servlet," whereas "format browse results" is responsible for formatting and creating dynamic html documents containing the results of the browsing activity. Similarly, the "gather ontology input" module extracts the ontology information provided by the user and routes it to either the "ontology creation" or "ontology evolution" module. The "format user feedback" module creates dynamic html documents containing the feedback from the system during the creation or evolution of the ontology.

b) Browse Module: The browse module provides mechanisms for viewing the ontologies that are currently stored in the repository. The user can search the repository through key words or select specific ontologies to view by name or id. From the user input, the "browse query generation" sub-module creates the appropriate query to retrieve one or more ontologies requested by the user. The "query execution" sub-module is responsible for establishing a session with the repository, executing the query, and retrieving the appropriate ontologies.

c) Logical Module: The logical module implements the ontology creation methodology and assists the user in creating or modifying an ontology. The module consists of "ontology creation", "ontology evolution" and "consistency checking" sub-modules. The ontology creation sub-module supports the basic steps outlined in our approach for creating ontologies and implements the heuristics and rules needed to create a consistent ontology. The ontology evolution sub-module facilitates the management and evolution of existing ontologies. As new information is discovered about a specific application domain, the corresponding ontology (ies) is extended to incorporate new knowledge. The "consistency checking" sub-module ensures that the ontology is consistent, by executing rules that check for inconsistencies between ontologies and within an ontology.

d) Physical Module: The physical module provides the interface to the ontology repository for storing and managing the ontologies. Once the user creates an ontology and checks for consistency, it is mapped to its physical representation (machine readable format) through the "internal representation" sub-module. Ontologies are represented using XML. The "repository management" sub-module provides facilities for organizing and storing the ontologies in secondary storage.

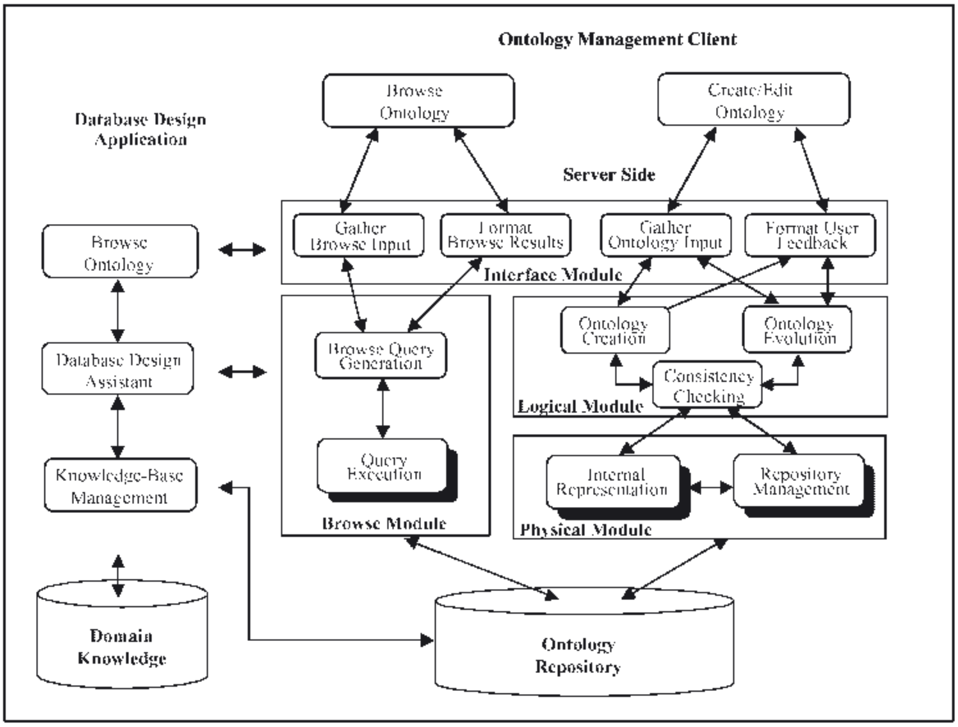

Figure 2. Ontology Management System Architecture

The Database Design Application is a proposed extension that will interface with the ontology repository and assist the designer. Using this tool, the designer obtains clarifications about concepts, terms and constraints in a particular application domain, resolves conflicts and performs other, relevant functions. This tool will have the capability to extract the domain knowledge contained in the ontology repository and create a knowledge base that could be used by the designer to gain a better understanding of the application domain.

4.1 Implementation

A prototype of the system is under development using Jess [Fr00], a java-based expert system shell, JavaBeans, and JSP (Java Server Pages). This development environment was chosen because it would make the system portable and easily usable on the world wide web. On the client side, the browsing and ontology creation functionalities are supported through java applets. On the server side, the interface, logical and physical modules are implemented using JSP technology. These modules utilize several java servlets for: gathering user input; generating queries in SQL for accessing and retrieving appropriate ontologies from the repository; supporting ontology creation and evolution; and formatting the results. Consistency checking is accomplished through rules written in Jess and the user is also consulted to gather additional information in order to resolve conflicts.

5 Conclusion

A methodology for the creation and adoption of domain ontologies for database design has been presented. The methodology is heuristic-based and focuses on identifying and defining the terms, properties, relationships and constraints needed to model an application domain. Meta-level rules help to ensure consistency. An architecture for the creation and management of domain ontologies will allow the methodology to be incorporated as part of a database design automation tool. Further research will implement the architecture to test and refine the methodology before it is tested on representative design problems. Future work will also include refining the consistency checking rules and empirical evaluation of the prototype by applying it to various application domains such as online auction, customer relationship management, etc.

Bibliography

[Al99] Altman, R. B., Bada, M., Chai, X. J., Carillo, M. W., Chen, R. O., Abernathy, N. F., RiboWeb: An Ontology-Based System for Collaborative Molecular Biology, IEEE Intelligent Systems and Their Applications, Vol. 14, No. 5, 1999, pp. 68 – 76.

[Em99] Embley, D., D.M. Campbell, Y.S. Jiang, Y.-K. Ng, R.D. Smith, S.W. Liddle, D.W. Quass, A conceptual-modeling approach to web data extraction, Data &Knowledge Engineering, 1999.

[Fr00] Friedman-Hill, E. Jess, the Expert System Shell, Sandia National Laboratories, Livermore, CA, 2000. URL: http://herzberg.ca.sandia.gov/jess

[Ja92] Jacobson, I., Christenson, M., Jonsson. P., and Oevergaard, G., Object-Oriented Software Engineering: A Use Case Driven Approach. Harlow: Addison-Wesley, 1992.

[KM99] Kedad, Z.. and Metais, E., Dealing with Semantic Heterogeneity During Data Integration, in Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., and Metais, E. (eds.), Conceptual Modeling ER'99, 18th Intl. Conference on Conceptual Modeling, Lecture Notes in Computer Science 1728, Paris, France, 15-18 November 1999. pp.325-339.

[Ll97] Lloyd-Williams, M., Exploiting Domain Knowledge During the Automated Design of ObjectOriented Databases, Proceedings of the 16th International Conference on Conceptual Modeling (ER'97), Los Angeles, 3-6 November, 1997.

[Lo99] Lopez, M. F., Gomez-Perez, A., Sierra, J. P., Sierra, A. P., Building a Chemical Ontology Using Methontology and the Ontology Design Environment. IEEE Intelligent Systems and Their Applications, Vol. 14, No. 1, 1999, pp. 37 – 45.

[Ne91] Neches. R., Fikes, R.E., Finin, T.. Gruber, T.R., Senator. T., and Swartout. W.R., Enabling Technology for Knowledge Sharing, AI Magazine, Vol.12, No.3, 1991, pp.36-56.

[PB99] Perez, A. G. and Benjamins, V. R., Overview of Knowledge Sharing and Reuse Components, Proceedings of IJCAI-99 Workshop on Ontologies and Problem Solving Methods (KRR5), 1999. Stockholm, Sweden.

[St01] Staab, S., Studer, R., Schnurr, H., Sure, Y. Knowledge Processes and Ontologies, IEEE Intelligent Systems and Their Applications, Vol. 16, No. 1, 2001, pp. 2 – 10.

[Sw99] Swartout, A.T., Ontologies, IEEE Intelligent Systems and Their Applications, Vol. 14, No. 1, 1999, pp. 18 - 19.