2002

Ozone: An Insulating Layer Between Ontologies, Databases and Object Oriented Applications

Yiannis Stavroulas
National Technical University of Athens, ystavroulas@telecom.ntua.gr

Theodora Varvarigou
National Technical University of Athens, dora@telecom.ntua.gr

Katerina Tsiara
National University of Athens, tsiara@telecom.ntua.gr

Katerina Tsiara
Agricultural University of Athens, tsiara@telecom.ntua.gr

OZONE: AN INSULATING LAYER BETWEEN ONTOLOGIES, DATABASES AND OBJECT-ORIENTED APPLICATIONS

Yiannis Stavroulas, Theodora Varvarigou, Yiannis Kouroupis, Katerina Tsiara

Department of Electrical and Computer Engineering, National Technical University of Athens
Athens, 157 73, Greece.
{ystavroulas, dora, gkou, tsiara}@telecom.ntua.gr

ABSTRACT

Recent research shows that ontologies are a prominent tool for the semantic integration of heterogeneous data sources. However, in existing ontology-based systems the ontologies are tightly coupled with the rest of the system components. As a result, large parts of the system have to be developed in a logic programming language, typically used in describing ontologies, and adhere to the ontological knowledge model and representation. This eventually impedes the use of ontologies in industrial integrated systems. In this paper, we present an architecture that isolates the ontology-based components, waives the representation and programming language constraints and simplifies the knowledge model that components outside the ontology have to be aware of. The architecture makes it possible to access the ontological information and the federated data using exclusively object-oriented structures and interfaces. We show that it allows new databases to easily join the federation by implementing a standard database interface. The architecture has been implemented and evaluated in the field of information retrieval for e-commerce. We review the principal results and limitations of this case study.

INTRODUCTION

We have developed Ozone in the context of MKBEEM[1], a project on multi-lingual, natural language-based e-commerce. From the database integration point of view, MKBEEM is a web-based information retrieval system. Its data sources are relational databases, which the system queries to retrieve information. There is no interest in insert and update operations, as the flow of information is always from the databases to the system. An additional complexity factor is that the system has to support natural language access. We explain how this requirement affects the source integration effort.

The purpose of this paper is to present the motivation and results of our work, without covering all the technical detail. In the remainder of this section, we define the context of our work by giving a brief overview of MKBEEM. In the second section, we review the current approaches to ontology-based database integration and point out their shortcomings that motivated this work. In the third section, we present Ozone in some detail and in the fourth we briefly describe the integrated system. The fifth section summarises our conclusions and discusses some possibilities for the extension of this work.

The scientific and technical aim of MKBEEM is to create an open marketplace where any content or service provider that wishes to make their products available through it, can do so by implementing a simple interface to the system. The system should be flexible enough that nationality, language and

[1] MKBEEM has been partially funded by the IST programme of the European Union, under contract number IST-1999-10589.

data organisation restrictions do not apply. The system's functionality is built around several key concepts:

- *Natural Language Interface.* In order to promote user friendliness and ease of use, the system accepts queries expressed in human language.
- *Multilinguality.* All the system's components are language-independent. Customers are able to select one of a number of supported languages (in our implementation English, French and Finnish) upon entry to the system, and interact exclusively in their language of choice from then on. In particular, care was taken to ensure that the system's answers to customer requests would not be biased to favour providers using the same language.
- *Homogeneity.* All product offerings should appear the same to the customers, regardless of their source (i.e. the different content providers).
- *Pan-European Scale.* The system must respect cultural and, most importantly, juridical particularities that depend on the nationality of the peers of a commercial transaction. Such differences range from cataloguing to applicable taxes and product returns legislation.

The prototype system carries out three main functions: answer a customer's requests for products or information, support the management of the system's product catalogues by the providers, and perform trading transactions between a customer and one or more providers. Only the first one will be described briefly here, as the other two are not relevant to this paper.

The customer request scenario evolves in three phases. In the first phase, the user must convey to the system the kind of product or service he looks for. This can be done either by browsing through the system's catalogues, or by describing the desired product in natural language. In the second phase, the system dynamically generates a form based on the ontological description of the desired product or service. Any constraints on the product's properties entered in natural language in the previous phase, are pre-filled in the form. The user inspects and fills in the form, then submits it to the system for processing. In the third phase, the system locates providers that offer the desired product, converts the form to a provider-specific query for each of them, executes the queries and returns the results.

CURRENT APPROACHES AND MOTIVATION

In the database integration literature there are three main approaches to the integration problem. The global schema approach [Batini et al., 1986], [Czejdo et al., 1987], [Dayal & Hwang, 1984], [Elmasri & Navathe, 1984], [Koh & Chen, 1993] solves the database integration problem by relying on a unified global schema and ad-hoc global-to-local schema mappings. The multi-database language approach [Pitoura, 1997], [Krishnamurthy et al., 1991], [Litwin et al., 1990], [Chen et al., 1993] delegates much of the integration task to the actual users, providing them with a flexible multi-database language, in which all local schemas are translated. The ontology-based approach [Hammer & McLeod, 1996], [Kahng & McLeod, 1996], [Wiederhold, 1994] [Mena et al., 1996], [Ullrich et al., 2001], [Wand & Weber, 1990] uses ontologies in the place of the global schema and maps them to the local schemas accordingly.

Ontologies constitute a competent solution to many database integration problems. However, the decisive factor for their use in the context of our system is the desired support for natural language. MKBEEM is designed to be a multilingual e-commerce portal, offering access to various content providers (CPs). Such a system faces three challenges:

- To integrate the various, possibly heterogeneous, CP systems
- To mediate between the CP systems and the customer
- To support user requests in natural language

This perspective is biased to differentiate between its front and back ends, i.e. the customers' domain and the providers' domain. This differentiation is largely artificial. Its only substantial justification is the flow of information, which is always from the CP systems to the customer. If we choose to ignore it, the above three challenges collapse into one: to integrate a number of information systems, each of which has its own knowledge model and data representation. In particular, each CP system uses its own database and corresponding schema, while the customer uses his native language, world knowledge and common sense.

This more abstract view provides an intuitive justification of the selection of ontologies as the integration mechanism. The global schema and multi-database language approaches are not adequate in this context, as they are specifically targeted to database systems. Both assume the local systems to have entity-relationship or at most object-oriented data models. This assumption clearly does not hold for the customers' system, where queries are expressed in human language and the underlying knowledge model is far too complex to be expressed in an entity-relationship scheme.

The ontology-based approach, on the other hand, is based on a formalism with much greater expressive power. It is capable of expressing the complex knowledge model of humans, at least in restricted domains. Furthermore it can provide the reasoning services that natural language understanding modules require, through the use of inference engines. There has been extensive research on the application of ontologies to the database integration problem, as well as to natural language understanding [Dahlgren, 1995]. Hence the ontology-based approach is ideal for a system that involves both these domains.

To sum up, the comparative advantage of ontologies in the context of MKBEEM is that they can contain all the necessary world knowledge in a single resource. They are used both for the integration of the various content sources and for the processing of human language. The advantage of having all this information encoded in a single resource is tremendous; in such a system, the cost and feasibility of creation and maintenance of semantic resources are often decisive success factors. This property distinguishes them from other database integration approaches and makes ontologies an excellent solution for any system where linguistic knowledge and database metadata must co-exist.

The State of the Art in Ontology-based Information Integration

The potential of ontologies as an integration medium in information retrieval systems has been recognised and exploited in several systems. We review here some recent successful examples, which by no means constitute an exhaustive list.

Ontobroker [Fensel, 1998], [Decker, 1999] is a WWW information retrieval system based on ontological tagging of HTML pages. The system offers a graphical query interface through which the users can input queries expressed in ontological terms. It is implemented in Frame-Logic and includes an inference engine, used for query and results processing. The authors acknowledge the problem of tagging the web pages and offer a simple annotation scheme and a supportive framework that allows annotations to be added incrementally, in a stepwise manner.

InfoSleuth [Bayardo, 1997], [Jacobs, 1996] is an agent-based system for information retrieval from structured or semi-structured sources. It has an extended agent model, in which agents communicate with messages, with the help of a dedicated service ontology. As in Ontobroker, the users express their queries in ontological terms, using terms from a number of domain ontologies. The agent layer of the system is implemented primarily in Java, but also uses LISP, CLIPS and LDL++ (a database language based on formal logic, with object-oriented extensions).

OBSERVER [Mena, 2000] uses multiple ontologies to retrieve information from Geographic Information Systems (GIS), emphasising on the inter-operation of multiple ontologies. It has a distributed architecture consisting of *component nodes*, where each component contains its own data repository, ontologies and query processor. Co-operation between nodes is managed by their ontology

servers through a central component used for the inter-operation of ontologies. The system uses CLASSIC, a language based on description logic.

Representation Heterogeneity

It is well understood that the integration in the semantic level of the system's heterogeneous data sources can be achieved by means of domain ontologies, in particular with the provision of a mapping between ontological concepts and each data source's schema. However, heterogeneity appears not only in the semantic level but also at the level of data representation; at this level, the ontologies further complicate the problem rather than help solve it.

In a system where ontologies are used to integrate heterogeneous databases in order to power a commercial web-based application, three data representation formalisms are likely to co-exist. On one hand, the databases use their own entity-relationship data representation, usually some dialect of SQL. On the other hand, the ontologies are likely to be encoded in a knowledge representation language. Finally, the vast majority of modern applications are written in object-oriented languages and thus rely on an object-oriented model of the world.

It is clear that the efficient transformation of data and metadata between these three representation formalisms is as important as the semantic integration itself. This constitutes the problem of representational integration. In the remainder of this paper we describe object-oriented structures and mechanisms that allow efficient access to a federation of databases, which employs ontologies for its semantic integration. As the actual application is written in an object-oriented language, an obvious choice for the system's universal representation formalism is this very language. Other alternatives would be an SQL dialect, the ontology's knowledge representation language, or even another formalism different from all three.

One could argue that the best universal representation formalism candidate would indeed be the knowledge representation language. This argument is supported by the fact that this is the richest of the three co-existing representations and it has the capability to subsume the other two. In fact, there has already been much successful research in this direction. The key weakness of these approaches is that they are hardly suitable for application in the industry. They are based on logic programming languages. Although there are methods to interface the resulting modules' APIs with object-oriented modules, a substantial part of the application must still be developed in logic programming.

Opportunities for Improvement

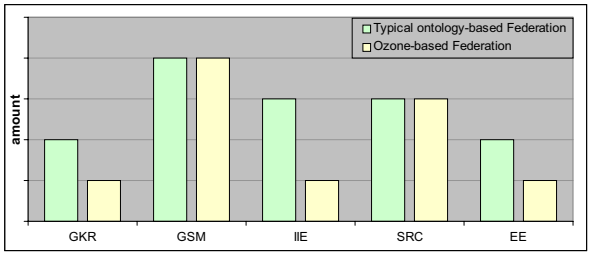
Although ontology-based integration techniques are in a mature stage, existing ontology-based integration systems fail to demonstrate a corresponding degree of maturity. Most are prototype, research-intensive systems, where the emphasis is on the ontologies and the inference components. As a result, they are developed in the greatest part on logic programming languages and make heavy use of complex knowledge representation concepts, tools and models.

The natural next step in the evolution of ontology-based integration systems is to move from this ontology-centric model to a component-based architecture, where the ontology-based integration subsystem constitutes a back-end module supporting one or more applications. The fact that ontologies carry with them their own programming paradigm and knowledge model, which takes the place of conventional data models, is highly inconvenient for application developers and software engineers. Thus the need arises for an architecture where ontologies can co-exist with conventional, widespread object-oriented platforms.

These considerations shape the guideline we followed in this work. There are four areas where we improve on present ontology-based information systems:

- *Insulation of the components based on description logic.* We provide a modular system model that limits the use of logic programming languages in the interior of the ontological module. The rest of the system can be purely object-oriented.
- *Hiding the complex ontological knowledge model.* We propose a minimal knowledge model, which is a simplification of the ontological knowledge model, with strong resemblances to the entity-relationship model. This model is much easier to understand and use, while still being adequate for the interactions of both the application developers and the content providers.
- *Provision of a tool for query formulation.* We propose an object-oriented structure that implements this knowledge model and offers the functionality for the formulation and manipulation of simple, form-based queries. It thus helps the application developers visualise and fill in form queries.
- *Database connectivity*. The interface to the databases is as straightforward as possible. Furthermore, we formalise a standard procedure that can be followed to perform the necessary ontology-schema mapping necessary to join the federation.

The model we describe is open and re-usable both on the front and back sides. Hence not only new databases can join the federation with minimal effort, but also new applications can be built on top of a pre-existing federation with minimal overheads.



GKR: Global Knowledge Required, amount of global knowledge that each component is supposed to have

GSM: Global Structures Maintained, globally, in order to allow information sharing and exchange between components

IIE: Initial Integration Effort, the amount of overhead effort required for each component, before it can join the system

SRC: Staticity of Relationships between Concepts

EE: Exchange Effort, the amount of computer time and user involvement required during the actual sharing and exchange

Figure 1: The impact of Ozone on the effort and knowledge required to join a database federation

In [Aslan, 1999] a comparison of various approaches to database integration is presented. In Figure 1 we visualise a comparison of Ozone against the standard ontology-based federation approach. Ozone improves in three aspects. First, it reduces the amount of knowledge about the word or other components that each component has to maintain, by introducing the simplified knowledge model. Second, it substantially reduces the overhead effort needed for new databases to join the federation, because it does not require them to understand complex knowledge models and use logic programming languages or formulas. And finally, it reduces the effort needed for an actual exchange, because it does not require the application and database components to express and manipulate queries in the cumbersome (for ontology-illiterate modules) form of ontological formulas.

THE OZONE ARCHITECTURE

Our system model is based on the Object-Oriented Ontology Access Layer, or Ozone layer. As Figure 2 illustrates, the Ozone layer is a three-way interface located between the databases of the content providers, the applications and the ontology server. It separates the system's back-end (i.e. the content

providers' databases) from its main bulk, that implements the business logic. Furthermore, it mediates between both the back-end and core system and the ontology server, thus insulating the system from the ontologies and imposing modularity to the system. It also adjusts the different representations; in Figure 2, the modules in shades of green use description logic-based representations, the orange modules use entity-relationship representations while the yellow modules are object-oriented.

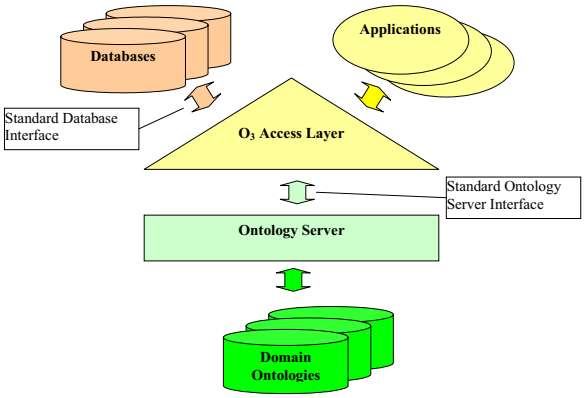


Figure 2: The Ozone Architecture

The Knowledge Model

Usually, ontologies use concepts or classes to model entities in the universe of discourse. Instance-of and other relationships, attributes, exceptions and axioms are used to model the properties, relations and behaviour of concepts [Fensel et al., 1998]. There is a variety of knowledge models employed by ontology creators, but typically concepts are organised into a taxonomy. Ontological concepts are much like classes in object-orientation, in that they have similarly defined subclass-of, superclass-of and instance-of relations. Like OO classes, ontological concepts or classes are formally defined as the set of their instances. In this paper we prefer the term concept rather than class, although it is perhaps less intuitive, merely to avoid confusion with OO classes.

The knowledge model we use in the object-oriented part of the system is based on these facts, while maintaining some correspondence with the entity-relationship model used in the databases. An ontology is defined as a taxonomy of concepts. Each concept comprises a concept name, a list of attributes and a list of relations to other concepts.

The distinction between attributes and relations here is somewhat artificial; it does not exist in the ontology's knowledge model. It is borrowed from the entity-relationship world. In the ontology knowledge model, concepts have only properties (or attributes or slots), which correspond to other concepts. In general, these must be handled as relations in our model. However, we choose to view relations to atomic concepts as attributes to keep correspondence with the entity-relationship model. For example, an ontological concept `Customer` may have, among others, the properties `Credit-card` and `Phone-number`. Assume also, that the concept `Credit-card` comprises several properties of its own, like the credit card type, number and date of expiry, while the concept `Phone-number` is atomic. In our knowledge model, the `Credit-card` property will be translated into a relation, while the `Phone-number` property will be translated into an attribute that takes a string value.

This knowledge model is essentially a hybridisation of the entity-relationship model, with the main difference being inheritance support. It is less expressive than the ontological knowledge model, but not less than the entity-relationship model. As a result, there is no overall loss in the system's expressiveness, but rather a redefinition of the boundaries between the most expressive parts of the system (the ontology server) and the less expressive ones (the applications and the databases), with Ozone being in the middle in terms of both physical location and expressiveness.

It should be noted though that this is not the knowledge model of the domain ontologies used in MKBEEM, which is substantially more complex. Apart from the integration of databases, the domain ontologies are meant to support the natural language querying system, which is more demanding in terms of reasoning capabilities and descriptiveness.

The Meta-Class

An object-oriented representation of this knowledge model de facto starts with a representation for concepts. The similarities between concepts and OO classes suggest that a class be used for each concept in the taxonomy. The Java reflection capabilities could be employed to create and compile the class at runtime, based on its ontological description. However, the resulting classes would differ only in variable members, as there is no way to read concept-specific code from the ontology and convert it to methods. This fact limits the usefulness of this solution, which still entails significant complexity and performance overheads.

Instead of this, we use a *meta-class* to model concepts. It is a meta-class from the knowledge model's point of view, as its instances are themselves ontological classes (i.e. concepts). Of course, from the OO point of view it is a regular class. Furthermore, each instance has a set of attributes and a set of relations to other concept instances. Formally, instances of the meta-class do not always represent ontological classes. In general an instance of the class represents a subset of the concept's instances, depending on the constraints on its attributes. This capability is essential for our intended use; it allows us to use instances of the class to represent queries.

Attributes are modelled as instances of an attribute class. The class provides support for typed, multi-valued attributes. It contains the attribute's name, maximum cardinality, type and measurement unit. There is the capability to define the range of values that are meaningful for an attribute, according to its semantics, as well as user-imposed constraints on them. The latter capability is useful for expressing queries.

Relations are handled similarly. A relation has a reference to the ontological relation that it instantiates. It further carries information on its allowed cardinality, direction and, of course, the source and target concepts.

The Interfaces

An important aspect of the system is the standardisation of a simple interface toward the databases. Our interface is the simplest possible, essentially containing only one method, that accepts an instance of the meta-class as a partially filled form containing attributes and relations of a specific concept, and returns all matching instances of the concept as instances of the meta-class.

In order to implement this interface, one needs to map global concepts and their attributes and relations to the local entities of the database. In order to further assist the interface developers, we have produced a formal procedure that can be used to perform the mapping. The procedure evolves in three stages. In the first, a concept is mapped to one or more entities of the database schema. In the second, the concept's attributes are mapped to fields in the database and in the third, the relations are mapped.

The interface to the applications is not presently standardised. An attempt to standardise this interface using only one application as a guide would be too risky. On the other hand, the interface to the ontology server is standardised and fairly simple. However this interface is of no particular interest, as it is implemented only once.

IMPLEMENTATION

The Ozone architecture has been applied in the context of MKBEEM and an experimental Ozone-based system has been implemented. In the following paragraphs we provide a brief description of MKBEEM and our implementation of the Ozone layer. Figure 3 shows the overall system architecture.

The *User Agent* is the front end of the system, being responsible for the management of the user interface, including the session management and the presentation of information in the appropriate human language.

The *Human Language Processing Servers* enable the translation of information from one human language to another, as well as the interpretation of human language sentences. The prototype system includes one server for each supported language. These servers work in co-operation with the Domain Ontology Server to convert natural language requests into ontological formulas.

The *Rational Agent* is the core of the system. It is the component that carries out the mediation between the customer and the providers. It is responsible for the processing of the customers' queries, including the dynamic creation of forms, the dispatching of queries to content providers and the grouping of the results. The functionality of the Rational Agent is split in two layers. The upper layer, called the Mediator, implements the business logic of the Rational Agent: it manages interaction with the user, the creation and execution of queries and the collection, processing and presentation of results. The Mediator is built on top of the Ozone layer, which facilitates the interaction with the content providers' database interfaces and the ontology server. Furthermore, the Ozone layer manages the conversion of ontological formulas resulting from natural language input, into form-based queries.

The *Catalogue Repository* stores the system's catalogues. The catalogues have two distinct roles. First, they offer a hierarchy of product categories, which the customers can browse. Second, they record the capabilities of each provider so that the Rational Agent can find out which of them supply a certain product.

The *Domain Ontology Server* is a knowledge base that provides information about the system's domain ontologies. Inference services are also available in the Domain Ontology Server.

The *CP Agents*, one for each content provider database that participates in the system, constitute the back end of the system. Each contains a map that relates ontological concepts to the proprietary database schema of the corresponding content provider. This map enables them to convert form-based queries to database-specific ones.

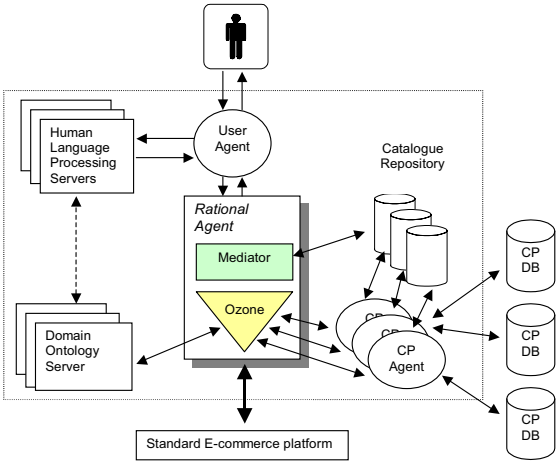
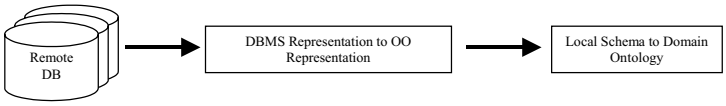


Figure 3: The Architecture of MKBEEM

Table 1 shows the transformation of a query as it traverses the system. Initially, the query is expressed in a natural language sentence. The HLP Server extracts the meaning of the sentence and represents in with an ontological formula. The formula is sent to the RA that consults the Ontology Server through the Ozone layer, in order to convert it to an object-oriented form. Then the form is sent to the appropriate CPAgents, converted to the local database query language and executed. The transformation of query results is illustrated in Figure 4. It is interesting to note here that the results do not need to be converted to an ontological formula representation throughout their lifetime.

Component	User Agent	HLP Server	Rational Agent	CP Agent	Remote DB
Representation	HL	OF	OO	OO	DB specific
Semantics	Common Sense	Ontology	Ontology	Remote Schema	Remote Schema

Table 1: Transformation of a query as it propagates through the system



Component	Remote DB	CP Agent	CP Agent	Rational Agent
Representation	DBMS	OO	OO	OO
Semantics	Remote Schema	Remote Schema	Ontology	Ontology

Figure 4: Transformation of data as it flows from a remote DB towards the application

CONCLUSIONS AND FUTURE WORK

In this paper we presented an architecture for ontology-based database federations, aimed at facilitating object-oriented information retrieval applications and reducing the effort required for new databases to join the federation. We introduced the Object Oriented Ontology Access Layer as a three-way interface that connects the application, the databases and the ontologies. The Ozone layer solves the problem of different representations throughout the system, by providing its own object-oriented representation of information, and mandating its mapping to and from the logic programming language used in the ontology. It effectively reduces the amount of ontological expertise required from database administrators and application developers by using a simplified hybrid knowledge model. Moreover, it insulates the ontologies from the data sources, so that no intervention to them is needed when the database schemas change, or a new database joins the federation.

Furthermore, we presented an overview of the experimental implementation of Ozone in the context of the project MKBEEM. This first experience verified that Ozone succeeds in substantially cutting down on the complexity of the development effort, without impairing the system's capabilities. It also allowed us to evaluate and formalise the interfaces of Ozone towards the ontologies and the databases, in order to be as simple as possible but still powerful enough to support a multi-lingual e-commerce application.

The next step in this research is to improve the interfaces of Ozone. The interface towards the applications can be generalised and formalised. This interface is currently biased towards our e-commerce application, however it would be interesting to assess it in other information retrieval contexts. Concerning the interface to the databases, it would be valuable to provide tools for assisting and, if possible semi-automating the process of joining the system. We are presently working in developing a database interface template module together with a customisation tool. The customisation tool is used by a database expert to output a map connecting the local database and the ontology. Adding this map and some configuration information to the interface template, we can get a fully functional interface module, customised to the particular database. Finally, the application of data mining techniques could be explored, in order to increase the degree of automation in the production of the mappings between domain ontologies and local data schemas.

REFERENCES

Aslan G, McLeod D: Semantic Heterogeneity Resolution in Federated Databases by Metadata Implantation and Stepwise Evolution. In The VLDB Journal (1999) 8. Springer-Verlag, 1999.

Batini C, Lenzerini M, Navathe S (1986) A comparative analysis of methodologies for database schema integration. ACM Comput Surv 18(4): 323–364 Comput 19(12): 10–18

Bayardo R J, Bohrer W, et. al., InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments.In Proceedings of the ACM SIGMOD International Conference on Management of Data, Vol. 26,2. ACM Press, New York, 1997.

Chen J, Bukhres O, and Elmagarmid A K (1993). "IPL: A multi-database transaction specification language," in Proceedings of the 1993 International Conference on Distributed Computing, 1993.

Corcho, O. and Gomez-Perez, A., "Solving Integration Problems of E-commerce Standards and Initiatives through Ontological Mappings", In: Proceedings of the Workshop on E-Business and Intelligent Web at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001), Seattle, USA, August 5, 2001.

Czejdo B, Rusinkiewicz M, Embley D (1987) An approach to schema integration and query formulation in federated database systems. In: Proceedings of the 3rd IEEE Conference on Data Engineering, 1987, Los Angeles, CA. IEEE Comp Society, pp 477–484

Dahlgren, K., "A Linguistic Ontology, International Journal of Human-Computer Studies, Vol.43, 1995, pp.809-818.

Dayal U, Hwang H (1984) View definition and generalization for database integration in a multi-database system. IEEE Trans Software Eng 10(6): 628–644

Decker S, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editor, DS-8: Semantic Issues in Multimedia Systems. Kluwer Academic Publisher, 1999.

Elmasri R, Navathe S (1984) Object integration in logical database design. In: Proceedings of IEEE Computer Society 1st International Conference on Data Engineering, 1984, Los Angeles, CA. IEEE Comp Society, 1984.

Fensel D, Stefan Decker, Michael Erdmann, and Rudi Studer. Ontobroker: The Very High Idea. In Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida, May 1998.

Hammer J, McLeod D (1993) An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. Int J Intelligent Coop Inf Syst 2(1): 51–83

Jacobs N, Shea R. The role of Java in InfoSleuth: Agent-based exploitation of heterogeneous information resources. In Proceeding of Intranet-96 Java Developers Conference, April 1996.

Kahng J, McLeod D (1996) Dynamic classificational ontologies for discovery in cooperative federated databases. In: Proceedings of the 1st International Conference on Cooperative Information Systems, June 1996, Brussels, Belgium. IEEE-CS Press

Koh JL, Chen ALP (1993) Integration of heterogeneous object schemas. In: Elmasri R, Kouramojian V, Thalheim B (eds) Proceedings of the 12th International Conference on Entity Relationship Approach, 1993, Lecture notes in CS, Vol 823, Springer, pp 297–314

Krishnamurthy R, Litwin W, Kent W (1991) Language features for IEEE interoperability of databases with schematic discrepancies. In: Clifford J, King R (eds) Proceedings of ACM SIGMOD International Conference on Management of Data, 1991, Denver Colo. SIGMOD Record 20(2): 40–49

Litwin W, Mark L, Roussopoulos N (1990) Interoperability of multiple autonomous databases. ACM Comput Surv 22(3): 267–293

Mena E, Illarramendi A, Vipul K, Sheth A. "OBSERVER : An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-existing Ontologies". In Distributed and Parallel Databases, 8. Kluwer Academic Publishers, 2000.

Pitoura E (1997) Providing Database Interoperability through Object-Oriented Language Constructs. In: Journal of Systems Integration, 7, 99 – 126. Kluwer, Boston, 1997.

Ullrich H , Purao S, Storey V. An Ontology for Classifying the Semantics of Relationships in Database Design. In: M. Bouzeghoub et al. (Eds.): NLDB 2000, LNCS 1959, pp. 91-102, 2001. Springer-Verlag, 2001.

Wand, Y., and Weber, R.. An Ontological Model of an Information System. IEEE Transactions on Software Engineering, Vol. 16, 1990, pp. 1282-1292.

Wiederhold G (1994) Interoperation, mediation and ontologies. In: Workshop on Heterogeneous Knowledge-Bases, 1994. W3, pp 33–48