Proc. of Concurrent Engineering (CE'2003), Special track Data Integration in Engineering, Madeira, Portugal, 26-30/07/2003, 217-224

An Automated Information Integration Technique using an Ontology-based Database Approach

Ladjel Bellatreche & Guy Pierra & Nguyen Xuan Dung & Dehainsala Hondjack
LISI/ENSMA - B.P. 401I9
86960 Futuroscope Cedex - FRANCE
{bellatreche, pierra, dung.nguyen-xuan, hondjack}@ensma.fr

ABSTRACT: Developing intelligent systems to integrate numerous heterogeneous data sources in order to give end users an uniform query interface is a great challenging issue. The process of constructing a global schema of the integrated system is usually done manually. This is because of the mechanisms of solving different conflicts existing in an heterogeneous context. Even more these data sources do not contain enough knowledge to help solving these conflicts and then generating the global schema. In this paper, we present a new approach to integrate heterogeneous sources based on a *priori approach* (ontology-based database). The originality of our approach is that each data source participating in the integration process contains an ontology that defines the meaning of its own data using an ontology model called PLIB. This approach ensures the automation of the integration process when (1) *all data sources have the same ontology*, and (2) *all sources reference a shared ontology, and each one can extent this ontology by adding new concepts*. Finally, we present integration algorithms for the previous cases.

1 INTRODUCTION

The advent of the Internet provides us the access to many autonomous and heterogeneous information sources. Information integration recently received a great attention due to many data management applications (example of data warehouses (Bellatreche, Karlapalem, & Mohania 2001)), e-commerce (Omelayenko & Fensel 2001) and engineering applications (e.g., a catalog management (Pierra 1997)). The purpose of information integration is to support seamless access to these data sources. A user accesses these sources in an uniform way, i.e., as if she is accessing one large database. Data integration is the problem of combining the data residing at different heterogeneous data sources, and providing to users an unified view of these data (this view is called global schema). Formally, a data integration system is a triple $I :< G, S, M >$, where G is the global schema (over an alphabet A_G) which provides a reconciled and a integrated schema, S is the source schema (over an alphabet A_S) which describes the structure of the set of sources participating in the integration process, and M is the mapping between G and S which establishes the connection between the elements of the global schema and those of the sources. Queries to a data integration system are posed in terms of the relations in G, and are intended to provide the specification of which data to extract from the virtual database represented by I. Two major architectures of the information integration are described: *materialized* (warehouse) and *virtual*. In the materialized approach, copies of information from several data sources are stored in a single database. In the virtual (mediator) architecture, a software called a mediator supports a virtual database (without storing data into a database), translates queries into source queries and synthesizes results and returns answers to a user query. Most of existing integration system dealing

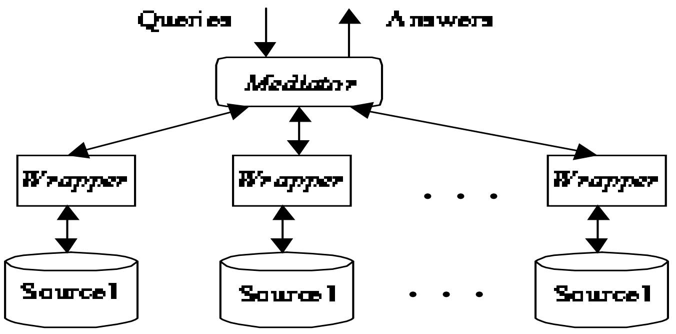

Figure 1: A Mediator Architecture

with the source heterogeneity use the mediation architecture (see figure 1), in which a mediator processes user queries by accessing source data. Two main concepts constitute this architecture (Ullman 1997): *wrappers* and *mediators*. A wrapper wraps an information source and models the source using a source schema. A mediator maintains a global schema and mappings between the global and source schemas.

Currently, there are two main approaches to integrate data and to answer queries without materializing a global schema: Global-as-View (GaV) or query-centric approach (TSIMMIS project (Chawathe, Garcia-Molina, Hammer, Ireland, Papakonstantinou, Ullman, & Widom 1994)) and Local-as-View(LaV) or source-centric approach (Manifold (Levy, Rajaraman, & Ordille 1996), Picsel (Franois Goasdoué, Lattès, & Rousset 2000)). In a GaV, the global schema is expressed as a view (a query) over the data sources. This approach facilitates the query reformulation by reducing it to simple execution of views in ordinary databases. However, changes in information sources or adding a new information source requires a database administrator (DBA) to revise the global schema and the mappings between the global schema and source schemas. Thus, GaV is *not scalable* for large applications. In the source-centric approach, each data source is expressed with one or more views over the global schema. Therefore, LaV scales better, and is easier to maintain than GaV because DBAs create a global schema independently of source schemas. Then, for a new source schema, the DBA only has to give (adjust) a source description that describes source relations as views of the global schema. In order to evaluate a query, a rewriting in terms of the data sources is needed. The rewriting queries using views is a difficult problem in databases. Thus, LaV has *low query performance* when users frequently pose complex queries.

The methodology of an integration process consists for four steps: (1) *Pre-integration*: analyzes schemas before integration to determine the integration technique, order of integration, and to collect additional information, (2) *schema comparison:* compares concepts and searches for conflicts and schemas properties, (3) *conforming the schemas :* solves schema conflicts, and (4) *merging and restructuring:* merges and restructures the schemas so they conform to certain criteria.

Any integration system should consider both integration at schema level (schema integration consists in consolidating all source schemas into a global or mediated schema that will be used as a support of user queries) and the data level (global schema population from the contents of data sources). Constructing an integrated view of data sources is a hard task because they (data sources) store different types of data, in various formats, different contexts, with different meanings, and different names. In order to provide an integrated view; solving different conflicts (*naming conflicts*, *scaling conflicts*, and *confounding conflicts*) is required [1]. Integration of heterogeneous data sources is facing to the meta-data management problem (Madhavan, Bernstein, Domingos, & Halevy 2002). For example, a data source may use EmployeeID to represent the employee number and another one uses Emp#. To solve these different conflicts, a higher level programming interface is required (Bernstein 2003). This interface can be done by using meta-data management (or model management). Without using meta-data management techniques, integrating heterogeneous data sources stays a difficult problem. Therefore, most of the existing integration systems solve these conflicts manually. Consequently, the integration process in these systems is done semi automatically (e.g., the schema and instances integration processes are done manually and automatically, respectively). This type of integration may compromise the *quality* of the integrated system (Bouzeghoub 2001). The circumstances of this manual integration is that the integrator (or the database administrator) do not have a rich ”knowledge” about the semantic and the schematic of the data sources.

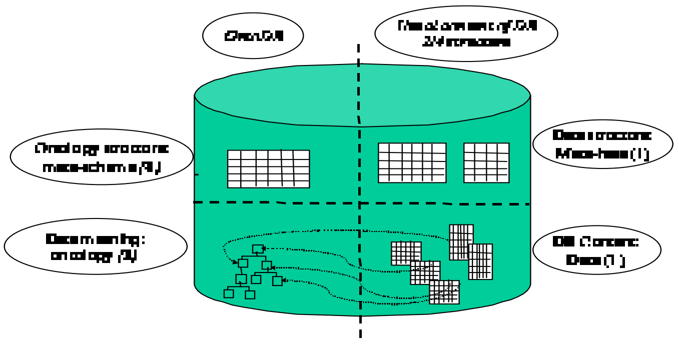
OntoDB
Ontology structure: meta-schema (4)
Data structure: Meta-base (1)
Data meaning: ontology (3)
DB Content: Data (1)

Figure 2: The Ontology-based Database Architecture

Recently, several researchers recommend the utilization of an ontology (describing the meaning of each data source) to capture different semantics and knowledge about data sources (Bernstein, Haas, Jarke, Rahm, & Wiederhold 2000).

In this paper, we present a novel approach of integrating data sources called *ontology-based database integration*. This approach is based on the a priori approach, where we assume that each data source contains an *ontology* as part of its schema as shown in Figure 2. The ontology of each data source is constructed based on the PLIB ontology model (Pierra 1997), (Pierra, Potier, & Sardet 2003). By using this

[1]These conflicts will be described in section 2.

approach, the process of integrating different sources [2] may be done automatically in the following cases:

1. Data sources use a same ontology (or a fragment of that ontology). This case is called *SameOnto*.

2. Data sources reference a shared ontology, and each source can extend it by adding new concepts and new properties [3] (this case is called *MapOnto*).

The use of the ontology-based database integration approach (a meta-data management technique) represents a great advance in developing intelligent integration systems by automating the major steps of the integration process as we will show in this paper.

The rest of this paper is organized as follows: in section 2, we describe the background of the integration problem in the context of heterogeneous sources, and we present an overview of PLIB ontology model that will be used as a basic support for our integration algorithms, in section 3, we present our integration approaches for the *SameOnto* and *MapOnto* cases. Related work is outlined in Section 4 and Section 5 concludes the paper.

2 BACKGROUND

To deal with data integration process in heterogeneous context, different categories of conflicts should be solved (Bernstein 2003), (Goh, Bressan, Madnick, , & Siegel 1999), (Wache, Vögele, Visser, Stuckenschmidt, Schuster, Neumann, & Hübner 2001): *naming conflicts* , *scaling conflicts*, and *confounding conflicts*. These conflicts may be encountered at *schema level* and *at data level*.

- *Naming conflicts* : occur when naming schemes of concepts differ significantly. The most frequently case is the presence of synonyms and homonyms.

- *Scaling conflicts*: occur when different reference systems are used to measure a value (for example price of a product can be given in dollar or in Euro).

- *Confounding conflicts :* occur when concepts seem to have the same meaning, but differ in reality due to different temporal contexts. For example, the weight of a person depends on the date where it was measured. Among properties describing a data source, we can distinguish two types of properties: context dependent properties (e.g., the weight of a person) and context non-dependent properties (gender of a person).

- *Representation conflicts :* arise when two source schemas describe the same concept in different ways. For example, in a source, student's name is represented by two elements *FirstName and LastName* and in another one it is represented by only one element *Name*.

2.1 The PLIB ONTOLOGY MODEL

Note that all sources contains ontology based on PLIB (Part Library) model. This model has been developed in 90's in order to exchange and integrate engineering component databases (Pierra 1990). Contrary to the existing ontology models (Gruber 1995), a PLIB ontology model has the following characteristics:

- *Conceptual*: each entity and each property are unique concepts completely defined, the terms (or words) used for describing them are only a part of their formal definitions.

- *Multilingual*: A global unique identifier (GUI) is assigned to each entity and property of the ontology. Textual aspects of their descriptions can be written in several languages. The GUI is used to identify exactly the similarities between properties (or between entities).

  Example 1 *Let S_1 and S_2 be two data sources constructed according the PLIB ontology model describing a Person. These two sources contain two properties with different names, let's say,* Person.Family_Name *and* Personne.Patronyme. *Based on their GUIs, two cases can be distinguished:*

  1. *These two properties have the same GUI (GUI1 as shown in Figure 3), for the integration system, these properties are identical (they represent the same thing, i.e., the family name of a person).*
  2. *They have different GUIs, for the integration system, they are different, even they have the same name (example of the property status in Figure 3).*

  One of the utilization of GUI is solving naming conflicts.

- *Modular*: an ontology can references another for different reasons: importing categories, properties without duplicating them.

- *Multi-representation*: Once a concept is defined, it can be associated to numerous representations, where each one is represented as a class (entity). The point of view of each representation is also a concept represented as a class by the ontology.

[2]The integration process includes consolidated ontology and schema generations and schema population

[3]The principle of existence of an ontology is to be shared

For example, we can define a class Person, the particular point of view of its study status (student view class) may be represented in the student representation class (Figure 4).

- *Consensual* : The conceptual model of PLIB ontology is based on an international consensus and published as international standards (IEC61630-4:1998, ISO13584-42:1998).

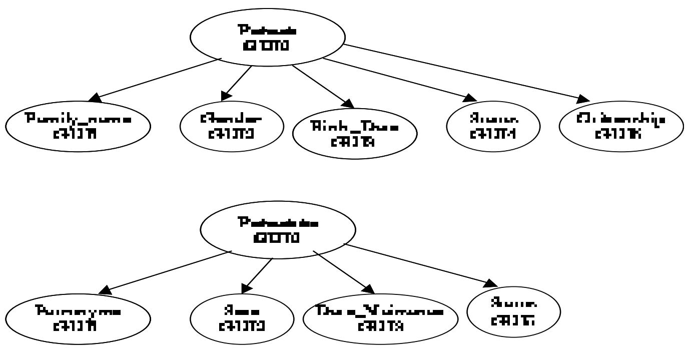

Figure 3: An example of solving naming conflicts in PLIB

Thus, a PLIB ontology consists of three categories of classes: *definition classes*, *representation classes*, and *view classes*. The definition class represents the beings of the areas of interests, with all their rigid properties. A property is rigid if it is essential to all its instances. The representation classes have additional properties depending on how the definition class are used for different purposes (or point of views) (Pierra 1993). The view classes capture the context of each particular representation class: each representation class shall reference a view class as its modeling context.

Example 2 *Consider a class definition of Person with the following rigid properties: Family name, Surname, Gender, Birthdate, etc. Based on this class definition, different representation classes of the class Person can be defined like Student representation class with additional properties like Registration number, discipline, Level, etc (see Figure 4). The view class defines the context corresponding to this representation class. For example it may contain properties like* name and location of the university. *The context is modeled in PLIB conceptual ontology model.*

The PLIB ontology model is defined and several tools were developed to create, validate, manage or exchange ontologies (such tools can be found at www.plib.ensma.fr).

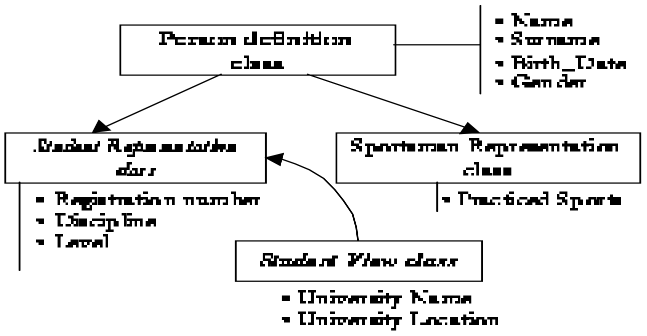

Figure 4: An example of different classes of the PLIB ontology model

2.2 ONTOLOGY-BASED DATABASES

Contrary the existing database structures (that contain two parts: data according to a logical schema and a meta-base describing tables, attributes, Foreign keys, etc.) an ontology-based database contains four parts : two parts as in the conventional databases plus the ontology definition and meta-base of that ontology. The relationship between the left and the right parts of this architecture associates to each instance in the right part its corresponding concepts defined in the left part. This architecture is validated by a prototype developed on Postgres (Dehainsala 2002).

3 ALGORITHMS FOR INTEGRATING ONTOLOGY-BASED DATABASE SOURCES

For the purpose of the discussion a stand alone PLIB ontology may defined as the quadruplet :
$O :< C, P, Sub, Applic >$, where:

- C is the set of the classes used to describe the objects of a given domain;

- P is the set of properties of the classes;

- Sub is is the subsumption function defined as $Sub : C \rightarrow 2^C$ [4], where for a class C of the ontology it associates its direct subsumed classes.

- $Applic : C \rightarrow 2^P$ is a function that associates to each ontology class those properties that are applicable (i.e., essential for each instance of this class).

  Example 3 *Figure 5 gives an example of an ontology that contains six classes (*$C = \{$Person, Student, LocalStudent, ForeignerStudent, Lecturer, AdminstrativeStaff$\}$*). A set of properties* $P = \{$*Name, Citizenship, BirthDate, ListOfProject, TuitionFee, Visa_Number*$\}$ *is assigned by the* $applic$ *function to each class. For instance,*

[4]We use the symbol 2^C to denote the power set of C.

the class Person *has the following properties: Name and Birthdate. The class* ForeignerStudent *has the property VisaNumber, etc. The subsumption function Sub defines specialization relationships between classes (for example, the class* Person*) subsumes the class* Student.

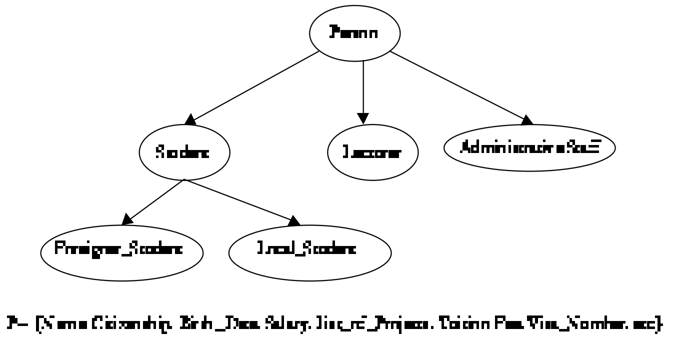
Person
Student
Lecturer
Administrative Staff
Foreigner_Student
Local_Student
P= {Name, Citizenship, Birth_Date, Salary, List_of_Projects, Tuition Fee, Visa_Number, etc}

Figure 5: An Example of an ontology and its different elements

An ontology-based database $OBDB$ is a quadruplet $OBDB :< O, I, Sch, Pop >$, where:

- O is an ontology ($O :< C, P, Sub, Applic >$);
- I is the set of instances of the database;
- $Sch : C \rightarrow 2^P$ associates to each ontology class the properties which are effectively used for describing class instances. This schema is chosen by the integrator for each leaf class (no-leaf classes are considered as abstract classes). This schema should ensure the following property: $\forall c \in C, Sch(c) \subset Applic(c)$ (only applicable properties may be used for describing class instances). It is computed by the system for non-leaf classes. We assume that a virtual table is created by the OBDB at each non-leaf class level as the projection on the applicable properties of these classes of all instances of its subclasses (subsumption representation);
- $Pop : C \rightarrow 2^I$ that associates to each class (leaf class or not) those own instances.

3.1 INTEGRATION WHEN ALL DATA SOURCES HAVE THE SAME ONTOLOGY

Let $S = \{SBO_1, SBO_2, ..., SBO_n\}$ be the set of data sources participating in the data integration process. Each data source SBO_i ($1 \leq i \leq n$) is defined as follows: $SBO_i =< O_i, I_i, Sch_i, Pop_i >$. The ontology O_i ($1 \leq i \leq n$) of each source is defined as a fragment of the common ontology O. It is defined as quadruplet $O_i :< C_i, P_i, Sub_i, Applic_i >$, where :

- $C_i \subseteq C$
- $P_i \subseteq P$
- $\forall c \in C_i, Sub_i(c) \subseteq Sub(c)$
- $\forall c \in C_i, Applic_i(c) \subseteq Applic(c)$

Integrating these n data sources, means finding an ontology, a schema and a population of the integrated system. Therefore the integration process $OInt$ is defined as triplet $OInt :< O_{OInt}, Sch_{OInt}, Pop_{OInt} >$. Now the question that we should answer is how to find the structure of each element of $OInt$?

- The ontology of the integrated system is O ($O_{OInt} = O$).
- The schema of the integrated system Sch_{OInt} is defined for each class c as follows:
  $Sch_{OInt}(c) = (\bigcap_A(\bigcap_B Sch_i(c'))) \bigcap Applic(c)$, [5]
  where $A = \{c' \in \{cl \cup Sub(c)\}\}$, and
  $B = \{i \mid c' \in C_i \wedge Pop_i(c') \neq \phi\}$.
- The population of each class of the integrated system Pop_{OInt} is defined as follows:
  $Pop_{OInt}(c) = (\bigcup_i proj_{sch(c)} Pop_i(c)$, where $proj$ is the projection operation as defined in classical databases.

3.2 INTEGRATION WHEN ALL DATA SOURCES REFERENCE A SHARED ONTOLOGY

This case differs from the previous one by the fact that each data source has its own ontology. But we assume that all the ontologies reference ”as much possible” a shared ontology $O :< C, P, Sub, Applic >$ in the following senses:

1. a local source contains a specific class if and only if a class with a same meaning does not exist in the shared ontology,
2. each class that does not belong to the shared ontology references explicitly its smallest subsumer within this ontology,
3. each property whose meaning is similar to a property of the shared ontology references explicitly this property.

[5] This defi nition ensures that instances of the integrated system are not expanded with null values to fi t with the more precisely defi ned instances. In place, only properties which are provided in all data sources are preserved. In some data sources may incur empty classes. These classes are removed from the set of classes used to compute the common provided properties.

Therefore, each source SBO_i maps the referenced ontology O to its ontology O_i. This mapping can be defined as follows: $M_i : C \rightarrow 2^{C_i}$, where $M_i(c) = \{$greatest classes of C_i subsumed by c$\}$. Contrary to the previous case, each data source SBO_i is defined as quintuple: $SBO_i =< O_i, I_i, Sch_i, Pop_i, M_i >$. To integrate the data sources in this scenario we should find the structure of the final integrated system $I^F :< O^F, Sch^F, Pop^F >$. Note that the structure of O^F is $< C^F, P^F, Sub^F, Applic^F >$. Each element of these structures is defined as follows:

- Integrated classes $C^F = C \bigcup_{(i \mid 1 \leq i \leq n)} C_i$,
- $P^F = P \bigcup_{(i \mid 1 \leq i \leq n)} P_i$,
- $\forall c \in C, \quad Sub^F(c) = Sub(c) \bigcup_{(i \in \{i \mid c \in C_i\})} Sub_i(c)$
- $Applic^F(c) = \begin{cases} Applic(c), \text{ if } c \in C \\ Applic(c_i), \text{ if } c \in C_i \wedge c \notin C \end{cases}$
- The population Pop^F of each class (c) is computed recursively using a post-order tree search. If c belongs to one C_i and does not belong to C, its population is given by: $Pop^F(c) = Pop_i^F(c)$. If c is a leaf of the integrated ontology tree; its population is defined as follows:

  $$Pop^F(c) = \begin{cases} \phi, \text{ if } c \notin C_i \ (1 \leq i \leq n) \\ \bigcup_{(i \mid 1 \leq i \leq n)} Pop_i(c), \text{ if } c \in C_i \end{cases}$$

  Otherwise (i.e., c is a no-leaf node of the shared ontology tree), $Pop^F(c)$ is defined as follows:
  $Pop^F(c) = \bigcup_{(c_j \in Sub^F(c))} Pop^F(c_j)$
- The schema of each class c of the integrated system is computed following the same principle as the population of c by considering leaf nodes and non-leaf nodes. If c is a leaf of the integrated ontology tree; its schema is defined as follows:

  $$Sch^F(c) = \begin{cases} \phi, \text{ if } c \notin C_i \ (1 \leq i \leq n) \\ \bigcap_{(i \mid Sch_i(c) \neq \phi)} Sch_i(c), \text{ if } c \in C_i \end{cases}$$

  Otherwise;
  $Sch^F(c) = \bigcap_{(c_j \mid c_j \in Sub^F(c) \wedge Sch^F(c_j) \neq \phi)} Sch^F(c_j)$

By using an ontology-based database approach for integrating heterogeneous sources, the task of describing formally the data integration process is simpler than in the previous systems and may be automated. It is important to notice that when all data sources use independent ontologies, the task of mapping these ontologies onto a receiver ontology may be done manually, but the integration process will be performed automatically as in the MapOnto case.

4 RELATED WORK

Several integration systems were proposed in the literature where we can categorize them into three major classes : (1) global as view systems (GaV), (2) local as view systems (LaV) and (3) systems based on semantic interoperability.

GaV systems In this category, we present three major systems: TSIMMIS (Chawathe, Garcia-Molina, Hammer, Ireland, Papakonstantinou, Ullman, & Widom 1994), Garlic (Roth, Arya, Haas, Carey, Cody, agin, Schwarz, Thomas, & Wimmers 1996) and SIMS (Arens & Knoblock 1993).
TSIMMIS (Chawathe, Garcia-Molina, Hammer, Ireland, Papakonstantinou, Ullman, & Widom 1994) is a mediator system developed at Stanford university. It aim is to simplify the extraction and integration of data from semi-structured data sources. TSIMMIS does not provide a mediator schema, but propagates all schemas of the components wrappers to the user. Data model heterogeneity is resolved using the semi-structured "Object exchange model" (OEM), a simple model with objects and object nesting. To resolve semantic conflicts between components, a dictionary service was proposed, but not implemented. The TSIMMIS was not developed to automates the integration process.

Garlic (Roth, Arya, Haas, Carey, Cody, agin, Schwarz, Thomas, & Wimmers 1996) is a project of IBM Research. It addresses large-scale multimedia information systems by considering specialized component systems to store and search for particular data types like image management systems. *Heterogeneity in schemas* is not considered.

SIMS ("Search in Multiple Sources") focuses on integration of heterogeneous databases and knowledge bases (Arens & Knoblock 1993). It uses a domain model for each integration application, described in LOOM, as semantic modeling language derived from KL-ONE. Sources are modeled according to this domain model. It uses an ontology to describe the domain about which information is stored in the information sources, as well the structure and contents of the information sources themselves. Wrapping sources is done manually.

LAV systems Information Manifold is a prototype developed by AT&T for integrating web-based data sources (Levy, Rajaraman, & Ordille 1996). Like TSIMMIS the global schema is virtual. The global view for all users called *world view* is expressed using the relational data model and class hierarchies. For each user access, Manifold identifies relevant sources and executes sub-queries to them. After that the user is responsible for cleaning overlapping information or performing object fusion mechanisms.

The integrated schema obtained by the previous integration systems is mapped to local sources by logical rules or query expressions specified by the designer. The mediators and wrappers used by these systems are not designed to define algorithms for resolving conflicts, or producing the global schema. The definition of the global schema and resolution of conflicts are done manually by the designer contrary to our approach.

Systems based on semantic interoperability COIN project (COntext Interchange) performs data integration based on logical axioms (Bressan, Goh, Levina, Madnick, Shah, & Siegel 2000). COIN has been designed to integrate structured and semi-structured data, where data sources are encoded into elevation axioms (for mapping values), context axioms (for representing context semantics), and conversion functions (for converting from one context to another). COIN did not build an integrated view, but it used a context mediator for querying and reconciling potential conflicts between data sources.

5 CONCLUSION

In this paper, we proposed an automated approach for integrating heterogeneous sources. We have presented a new integration technique called ontology-based database integration approach. The presence of an ontology modeled according to the PLIB ontology model in each database helps in capturing the knowledge about data, a schema, properties, etc. Therefore it contributes to the automation of the integration process contrary to the existing techniques. This automation is ensured in two cases: (1) all sources use the same ontology and (2) they reference a shared ontology.

In addition to its capability for automating the integration process of heterogeneous databases (note that a prototype of developing an ontology-based database is currently in progress in our laboratory), there are many other future directions that need to be explored. Some of the more pressing ones are:

- Evaluating this approach in a real situation and comparing it with the existing approaches in order to measure the quality of the integration schema.
- Extending the utilization of this approach to different application scenarios like schema evolution, schema exchange, schema matching.
- Considering the query optimization aspect to see how an ontology can be used for indexing query (semantic indexing).
- Providing a cost model to evaluate queries on a global schema on the integrated system. This cost model should take into account the ontology-based database structure (four parts).

REFERENCES

Arens, Y. & Knoblock, C. A. (1993, May). Sims: Retrieving and integrating information from multiple sources. *Proceedings of the International Conference on Management of Data (SIGMOD'1993)*, 562–563.

Bellatreche, L., Karlapalem, K., & Mohania, M. (2001). Some issues in design of data warehousing systems. In *in Developing Quality Complex Data Bases Systems: Practices, Techniques, and Technologies, Edited by Dr. Shirley A. Becker*. Idea Group Publishing.

Bernstein, P. (2003). Applying model management to classi-cal meta data problems. *in Proceedings of the 2003 CIDR Conference*.

Bernstein, P., Haas, L. M., Jarke, M., Rahm, E., & Wiederhold, G. (2000). Panel: Is generic metadata management feasible? *vldb*, 660–662.

Bouzeghoub, M. (2001, April). Ausing semantics to improve schema and data integration. *Proceedings of the International Workshop on Information Integration on the Web (Invited Talk*.

Bressan, S., Goh, C., Levina, N., Madnick, S., Shah, A., & Siegel, M. (2000, September). Context knowledge representation and reasoning in the context interchange system. *Applied Intelligence 13*(2), 165–180.

Chawathe, S. S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. D., & Widom, J. (1994, Marsh). The tsimmis project: Integration of heterogeneous information sources. *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, 7–18.

Dehainsala, H. (2002). Analyse et implmentation d'un modèle objet express dans une base de données relationnelle objet: Postgres. In *Mémoire d'Ingénieur - Stage effectué au LISI/ENSMA*.

Franois Goasdoué, F., Lattès, V., & Rousset, M. C. (2000, December). The use of carin language and algorithms for information integration: The picsel system. *International Journal of Cooperative Information Systems (IJCIS) 9*(4), 383–401.

Goh, C., Bressan, S., Madnick, E., , & Siegel, M. D. (1999). Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems 17*(3), 270–293.

Gruber, T. (1995). A translation approach to portable ontology specifi cation. *Knowledge Acquisition 5*(2), 199–220.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996, June). The world wide web as a collection of views: Query processing in the information manifold. *Proceedings of the International Workshop on Materialized Views: Techniques and Applications (VIEW'1996)*, 43–55.

Madhavan, J., Bernstein, P., Domingos, P., & Halevy, A. (2002). Representing and reasoning about mappings between domain models. *in Proceedings of the 18th National Conference on Artifi cial Intelligence (AAAI'02)*, 80–86.

Omelayenko, B. & Fensel, D. (2001, September). A two-layered integration approach for product information in b2b e-commerce. *Proceedings of the Second International Conference on Electronic Commerce and Web Technologies,* 226–239.

Pierra, G. (1990). An object oriented approach to ensure portability of cad standard parts libraries. *Proceedings of the European Computer Graphics Conference and Exhibition (Eurographics'90)*, 205–214.

Pierra, G. (1993). A multiple perspective object oriented model for engineering design. *in New Advances in Computer Aided Design & Computer Graphics*, 368–373.

Pierra, G. (1997, April). Intelligent electronic component catalogues for engineering and manufacturing. *Symposium on Glogal Engineering Networking (GEN'97), Special Session on Intelligent Electronic Catalogues*, 331–352.

Pierra, G., Potier, J. C., & Sardet, E. (2003). From digital libraries to electronic catalogues for engineering and manufacturing. *International Journal of Computer Applications in Technology (IJCAT) 18*, 27–42.

Roth, M. T., Arya, M., Haas, L., Carey, M., Cody, W., agin, R., Schwarz, P., Thomas, J., & Wimmers, E. (1996, June). The garlic project. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 557–557.

Ullman, J. D. (1997, January). Information integration using logical views. *Proceedings of the International Conference on Database Theory (ICDT), Lecture Notes in Computer Science 1186*, 19–40.

Wache, H., V¨ogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & H¨ubner, S. (2001, August). Ontology-based integration of information - a survey of existing approaches. *Proceedings of the International Workshop on Ontologies and Information Sharing*, 108–117.