To appear in: Proc. of Database and Expert Systems Applications - DEXA '04

# An a Priori Approach for Automatic Integration of Heterogeneous and Autonomous Databases

Ladjel Bellatreche Guy Pierra Dung Nguyen Xuan
Dehainsala Hondjack Yamine Ait-Ameur

LISI/ENSMA Téléport2 - 1, Avenue Clément Ader
86960 Futuroscope - FRANCE
E-mail : (bellatreche, pierra, nguyenx, hondjack, yamine)@ensma.fr

**Abstract.** Data integration is the process that gives users access to multiple data sources though queries against a global schema. Semantic heterogeneity has been identified as the most important and toughest problem when integrating various data sources. Several approaches were proposed to deal with this problem. These approaches can be classified using three criteria: (1) *data representation* which means whether data of sources will be materialized in a warehouse at the integrated system level or accessed via a mediator, (2) *the sense of the mapping* between global and local schemas (e.g., Global as View, Local as View) and (3) *the nature of the mapping (manual, semi automatic and automatic*). Mapping is manual each time when ontologies are not used to make explicit data meaning. It is semi automatic when ontology and ontology mapping are defined at integration level. In this paper, we propose a fully automatic integration process based on ontologies. It supposes that each data source contains a conceptual ontology that references a shared ontology. The mappings between a local ontology and the shared ontology is defined at *database design time* and also *embedded in each source*. This approach is implemented using PLIB-based ontologies (officially ISO 13584). It is assumed that there exists a domain ontology, but each data source may extend it by adding new concepts and properties. Therefore the shared ontology is referenced when ever it is possible. This integration approach was developed for automatic integration of component databases. It is currently prototyped in various environments including OODB, ORDB, and RDB.

## 1 Introduction

The overwhelming amount of heterogeneous data stored in various data repositories emphasizes the relevance of data integration methodologies and techniques to facilitate data sharing. Nowadays integrating heterogeneous and autonomous data sources represents a significant challenge to the database community. The availability of numerous sources increases the requirements for developing tools and techniques to integrate these sources. Data integration is the process by which several autonomous, distributed and heterogeneous information sources (where each source is associated with a local schema) are integrated into a single data source associated with a global schema. Data integration recently received

a great attention due to many data management applications : examples are Peer-to-Peer data [1], Data Warehouse [2], and E-commerce [15].

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 an integrated schema, $S$ is a set of source schemas (over an alphabet $A_S$) which describes the structure 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 $I$.

Various integration systems have been proposed in the literature [3, 11, 5, 18, 13]. Their fundamental problem is their inability to integrate automatically at the meaning level several *heterogeneous* and *autonomous* data sources. In the first generation of integration systems (e.g., TSIMMIS [5]), data meaning was not explicitly represented. Thus, concept meaning and mapping meaning were manually encoded in a view definition. The major progress toward automatic integration resulted from the explicit representation of data meaning through ontologies [22]. Various kinds of ontologies were used, either linguistic [4] or more formal [10]. All allowed some kind of partially automatic integration under expert control. In a number of domains, including Web service, e-procurement, synchronization of distributed databases, the new *challenge* is to perform fully automatic integration of autonomous databases. We claim that: *if we do not want to perform human-controlled mapping at integration time, this mapping shall be done a priori at the database design time*. This means that some formal shared ontologies must exist, and each local source shall embedded some ontological data that references explicitly this shared ontology. Some systems are already developed based on this hypothesis: Picsel2 [18] project for integrating Web services, the COIN project for exchanging for instance financial data [7]. Their weakness is that once the shared ontology is defined, each source shall used the common vocabulary. The shared ontology is in fact a global ontology and each source is less autonomous.

Our approach gives more autonomy to various data sources. To achieve this goal:

1. each data source participating in the integration process *shall* contain its own ontology. We call that source an ontology-based database (OBDB).
2. each local source references a shared ontology.
3. local ontology may extend the shared ontology as much as needed.

Consequently, the automatic integration process involves two steps: automatic integration of ontologies and then an automatic integration of data.

The context of our work is the automatic integration of industrial component databases [17]: we have already prototyped several implementation of the proposed approaches in object oriented database, object-relational database, and relational database environments.

Our approach requires that the target domain is already modeled by a shared consensual (e.g., standard) ontology. We already contribute to the development of such ontologies at the international standardization level (e.g., IEC61630-4:1998). A number of other initiatives go to the same direction [18].

In this paper, we present a novel integration approach called *conceptual ontology-based database integration*. Contrary to linguistic ontologies, ontologies used in our integration process are formal (no synonymous), consensual, embedded with each data source (consequently it can be exchangeable), extensible using the subsumption relationship (each source may add whatever property or class). Like COIN [7] (where the ontology represents a contextual information of values), our ontology *also* represents the context of the ontology definition.

To the best of our knowledge, the proposed work is the first article that addresses the integration problem supposing that a conceptual ontology is embedded in each data source.

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, in section 3 we propose a classification of integration approaches that facilitates the position of our work from the previous work, in section 4 we present an overview of the ontology model that will be used as a basic support for our integration algorithms, in section 5, we present the concept of ontology-based database and its structure, in section 6; integration algorithms are presented, and section 7 concludes the paper.

The main contributions of this paper are:

1. A new classification of integration systems using three major criteria's: data representation, the sense of the mapping between global and local schemas, and (3) the nature of the mapping.
2. An integration approach based on a priori approach ensuring a fully automatic integration process and respecting the autonomy of each data source.
3. A new structure of storing data sources with their local ontology (ontology based database).
4. A well suited formal ontology model called PLIB.

## 2 Background

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 at data level (global population). Constructing a global schema from local sources is difficult because sources store different types of data, in varying formats, with different meanings, and reference them using different names. Consequently, the construction of the global schema must handle different mechanisms for reconciling both data structure (for example, a data source may represent in the same field first and last name, when another splits it into two different fields), for data meaning (for example synonymous, hynonymous).

Let $S = \{S_1, S_2, ..., S_n\}$ be a set of heterogeneous data sources that participate in the integration process. The main task of integrating these sources is identifying the equivalent concepts (and properties). To do so, different categories of conflicts should be solved. Goh et al. [7] suggest the following taxonomy: *naming conflicts*, *scaling conflicts*, *confounding conflicts* and *representation 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 measuring 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*.

## 3 A Proposed Classification for Integration Systems

It is very difficult to classify the previous data integration techniques. Most of the papers distinguish two major categories: *Local as View (LaV)* [6, 18, 13], and *Global as View (GaV)* [5]. Other contribution distinguish *single ontology*, *multiple ontologies*, and *shared ontology* [22]. Some other work focus on the place of data and distinguish *mediator approach* and *warehouse approach* [20].

### 3.1 Data Representation

This criteria specifies whether data of local sources are duplicated in a data warehouse or not. The data of the integrated system may be *virtual* (it remains in the local source like in TSIMMIS [5] and accessed through a mediator, or it may be *materialized* (duplicated).

### 3.2 The Sense of the Mapping between the Global and Local schemas

In GaV systems, the global schema is expressed as a view (a function) over 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, the global schema is defined as an ontology [18], independently of source schemas. 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 [12]. Thus, LaV has *low query performance* when users frequently pose complex queries.

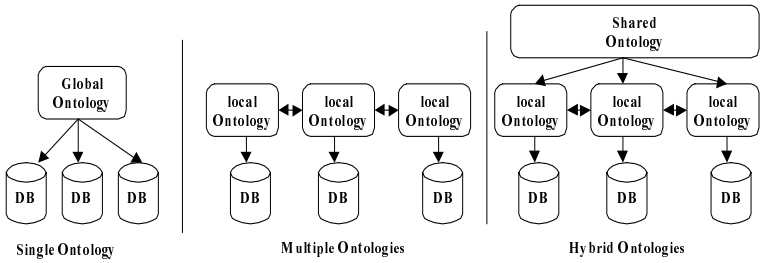


**Fig. 1.** Different Ontology Architectures

### 3.3 The Nature of the Mapping

This criteria specifies whether the mapping between the global schema and local schemas is done **manually**, **semi-automatic**, or **fully automatic**. The manual mapping is found in the first generation of integration systems that integrate sources represented by a schema and a population (i.e., each source $S_i$ is defined as : $< Sch_i, Pop_i >$ as in classical databases) and without explicit meaning representations.

The manual systems focus mainly on query support and processing at the global level, by providing algorithms for identifying relevant sources and decomposing (and optimizing) a global query into sub queries for the involved sources. The construction of the mediators and the wrappers used by these systems is done manually because their main objective focus on global query processing [4].

To make the data integration process (partially) automatic, explicit representation of data meaning is necessary. Thus most of the recent integration proposed approaches using ontologies [10, 4, 4, 18]. Ontologies are consensual and explicit representations of conceptualization [8]. Based on the way how ontologies are employed, we may distinguish three different architectures [22]: *single ontology methods*, *multiple ontologies methods*, and *hybrid methods* (see figure 1). In the single ontology approach, each source is related to the same global domain ontology (e.g., Lawrence et al. work [11] and Picsel [18] and [10]). As

a result, a new source cannot bring new or specific concepts without requiring change in the global ontology. This violate the *source autonomy* requirement (each source can operate independently). In the multiple ontologies approach (e.g., Observer, [14]), each source has its own ontology developed without respect of other sources. Then, inter-ontology meanings are defined. In this case the definition of the inter-ontology mapping is very difficult as different ontologies may use different aggregation and granularity of the ontology concept [22]. Hybrid approach has been proposed to overcome the drawbacks of single and multiple ontologies approaches, where each source has its own ontology, but all ontologies are connected by some means to a common shared vocabulary (e.g., KRAFT project [21]).

Any way, in all these approaches, ontologies and ontology mappings are defined at integration time. Therefore, they always request an human supervision, and they are only partially automatic. To enable automatic integration, semantic mapping shall be defined at database design time. This means that there shall exist some shared ontology, and more ever, each local source contains ontological data that refers to the shared ontology. Some systems have already been proposed on that direction such as *Picsel2* [18], *COIN* [7]. But to remain automatic, these systems do not provide autonomy to each data source in adding concepts and properties.

Our OBDB approach belongs to this category, but we also allow each data source to make its own extension in the shared ontology.

| **Integrated Systems** | **Data Representation** | **Sense of mapping** | **Nature of mapping** |
|---|---|---|---|
| TSIMMIS | Virtual | GaV | Manual |
| PICSEL | Virtual | LaV | Semi Automatic |
| OBSERVER | Virtual | GaV | Semi Automatic |
| MANIFOLD | Virtual | LaV | Semi Automatic |
| MOMIS | Virtual | GaV | Semi Automatic |
| COIN | Virtual | LaV | Automatic |
| KFRAFT | Virtual | LaV | Semi Automatic |
| ZURICH'Project [10] | Warehouse | LaV | Semi Automatic |
| PICSEL2 | Virtual | LaV | Automatic |
| OBDB | Virtual/warehouse | LaV | Automatic |

**Table 1.** Integrated Systems Classification

## 4 The PLIB ontology model

To describe the meaning and the context of each data source, we can use any ontology language like OWL, PSL, DAML+OIL, Ontolingua, etc. In this paper

we use the PLIB ontology model because of number of domain ontologies based on this model already exist or are emerging (e.g., IEC, JEMIMA, CNIS, etc. see http://www.plib.ensma.fr) and also because it was precisely designed to promote data integration [16]. In such a context, providing an approximate integration (as it might be obtained using linguistic ontologies) is worse than providing no answer at all.

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 globally unique identifier (GUI) is assigned to each entity and property of the ontology. Textual aspects of their descriptions can be written in several languages (French, English, Japanese, etc.). The GUI is used to identify *exactly* one concept (property or entity) and automatic mapping.
- **Modular**: an ontology can reference another one for importing entities and properties without duplicating them. Thus providing for autonomy of various sources that do reference a shared ontology.
- **Consensual** : The conceptual model of PLIB ontology is based on an international consensus and published as international standards (IEC61630-4:1998, ISO13584-42:1998) (for more details see [16]).
- **Unambiguous** : Contrary to linguistic ontology models [16], where partially identical concepts are gathered in the same ontology-thesaurus with a similarity ratio (affinity) [4, 19], each concept in PLIB has with any other concepts of the ontology well identified and explicit differences. Some of these differences are computer-interpretable and may be used for processing queries, e.g., difference of measure units, difference of evaluation context of a value.

### 4.1 Automatic Resolution of Naming Conflicts in PLIB

One of the utilization of GUI is solving naming conflicts (due to synonymous and hynonymous) as shown in the following example.

*Example 1.* Let $S_1$ be a source referencing the PLIB ontology model describing a Person (Figure 2). This source has the autonomy to use different names of its attributes (for example, it may use Nom instead of name). For the integrated system, these two attributes (properties) are similar because they have the same GUI. More generally, if several sources use different names; we can identify easily whether they are different or identical using the following procedure:

1. These two properties have the same GUI, for the integration system, these properties are identical (they represent the same thing, i.e., the family name of a person), even they have different names.
2. They have different GUIs, for the integration system, they are different, even they have the same name.

Note that unlike other integration systems based on linguistic ontologies where affinity measurements and thresholds are used to compute the similarity between concepts [4, 19], the orthogonality of PLIB ontologies and the use of GUI make the resolution of naming conflicts *deterministic* and *fully automated*.

### 4.2 PLIB Class Properties

The PLIB ontology model is represented by a tree of classes, where each class has its own properties. Two types of properties are distinguished: *rigid properties* (a rigid property is a property that is essential to all instances of a class [9]) and *role dependent properties* that may or not hold or exist according to a role in which an entity is involved (for example, salary property of a class Person is a role dependent property because it exists if the person is an employee, and it may exist several times if the person play several times the same role). In a database schema, a Person may have a salary property but this is based on a context that shall be explicit at the ontological level (e.g., the company where the person is employed, the date and the currency of the salary).

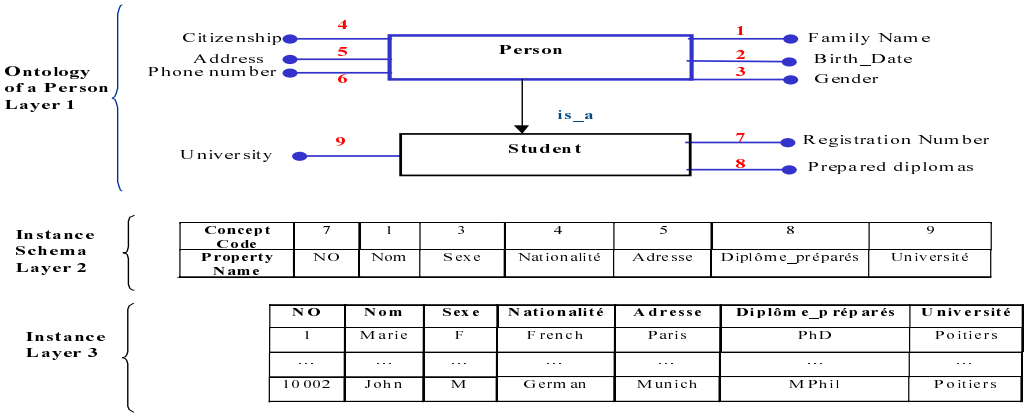


**Fig. 2.** An Example of Specializing a Global Ontology

### 4.3 Extension Possibilities Offered by PLIB

When a PLIB ontology model is shared between various sources (because these sources commit on the ontological definitions that were agreed and possibly standardized (e.g., IEC, JEMIMA), each source remains autonomous and may extend the shared ontology using subsumption relationship. Two relationships are distinguished: specialization (is-a) and case-of.

***Is-A Relationship*** A local source can extend the shared ontology by specializing class(es) of the shared ontology. Through this relationship, properties are inherited. Figure 2 considers a shared ontology describing a Person with six properties Family name, Birth date, Gender, Citizenship, Address and Phone number. A local source may specialize this shared ontology in order to define its own ontology describing Student with three other properties.

***Case-Of Relationship*** In this case properties are not inherited but may be explicitly (and partially) imported. Figure 4 shows an extension of the shared ontology Person using the case-of relationship. The local ontology PhD Student imports some properties of the shared (Family name, Religion, Citizenship, Address). Note that this local ontology does not import some properties of the shared ontology like Birth date, and Birth Citizenship. To respond to its need, it adds other properties describing a PhD student like registration number, Advisor and Thesis subject.

The PLIB ontology model is completely stable and several tools have been already been developed to create, validate, manage or exchange ontologies (such tools can be found at PLIB home site: www.plib.ensma.fr).

## 5 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-model 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 meaning defined in the left part. This architecture is validated by a prototype developed on Postgres.

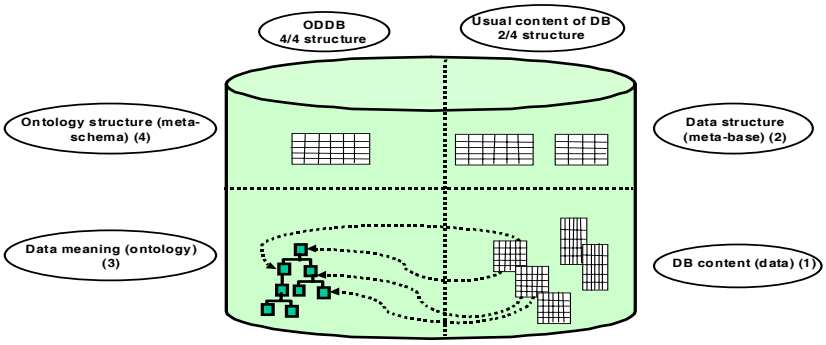


**Fig. 3.** The Ontology-based Database Architecture

### 5.1 Formal Definition of an Ontology-based Database

Formally, a PLIB ontology may be defined as the quadruplet : $O :< C, P, Sub, Applic >$, where:

- $C$ is the set of the classes used to describe the concepts of a given domain (like travel service [18], equipment failure, etc);
- $P$ is the set of properties used to describe the instances of the C classes. Note that it is assumed that $P$ defines a much greater number of properties that are usually represented in a database. Only a subset of them might be selected by any particular database [1].
- $Sub$ is the subsumption (is-a and case-of) function (Figure 2, 4 defined as $Sub : C \rightarrow 2^C$ [2], where for a class $c_i$ of the ontology it associates its direct subsumed classes [3]. $Sub$ defines a partial order over $C$.
- $Applic$ is a function defined as $Applic : C \rightarrow 2^P$. It associates to each ontology class those properties that are applicable (i.e.,rigid) for each instance of this class. Applicable properties are inherited through is-a subsumption and partially imported through case-of subsumption.
  Note that as usual ontological definitions are intentional: the fact that a property is rigid for a class does not mean that value will be explicitly represented for each instance of the case. In our approach, this choice is made among applicable properties at the schema level.

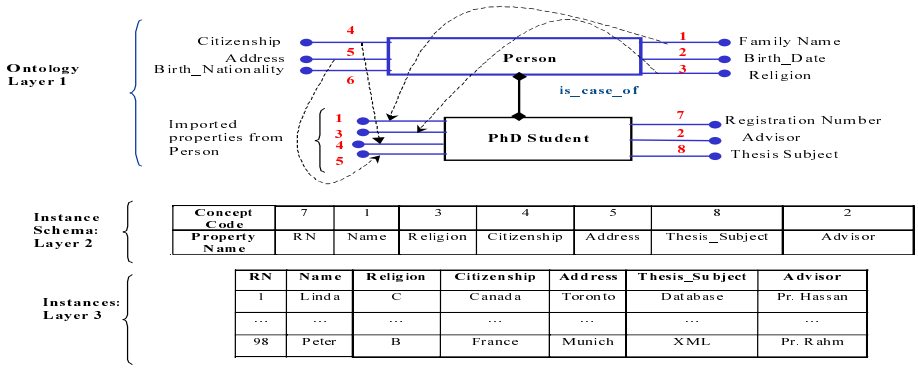


**Fig. 4.** An Example of Extending a Global Ontology

[1] A particular database may also extend the $P$ set
[2] We use the symbol $2^C$ to denote the power set of $C$.
[3] $C_1$ subsumes $C_2$ iff $\forall x \in C_2, x \in C_1$.

*Example 2.* Figure 4 gives an example of an ontology with two classes $C = \{Person$ and $PhD\ Student\}$. Let $P = \{Family\_Name, Citizenship, Birth\_Date, Religion, Address, Birth\_Citizenship, Registration\_Number, Advisor, Thesis\_Subject\}$ be set of properties that characterize these classes. Properties in $P$ will be assigned to classes of the ontology (therefore, each class will have its own rigid properties). The subsumption function $Sub$ defines a case-of relationship between classes (for example, the class *Person* subsumes the class *Phd Student*).

An ontology-based database $OBDB$ allows to record together within ontology a set of instance of ontology classes. Thanks to the subsumption relationship, an instance belongs to several classes. For the purpose of simplicity, we assume that each instance belongs to exactly one leaf class (non-leaf classes are "abstract"). Formally, an $OBDB$ is a quadruplet $< O, I, Sch, Pop >$, 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 $c_i$ of $C$ the properties which are effectively used to describe the instances of the class $c_i$. $Sch$ has two definitions based on the nature of each class (a leaf or a no-leaf class).
  - *Schema of each leaf class* $c_i$ is explicitly defined. It shall only ensure the following:
    $$\forall c_i \ \in C, Sch(c_i) \subset Applic(c_i) \tag{1}$$
    (Only applicable properties may be used for describing class instances of $c_i$).
  - *Schema of a no-leaf class* $c_j$ is computed. It is defined by the intersection between the applicable properties of $c_j$ and the intersection of properties associated with values in all subclasses $c_{i,j}$ of $c_j$.
    $$Sch(c_j) = Applic(c_j) \bigcap (\bigcap_i Sch(c_{i,j}) \tag{2}$$
    An alternative definition may also be used to create the schema of a no leaf class where instances are completed with null values:
    $$Sch'(c_j) = Applic(c_j) \bigcap (\bigcup_i Sch(c_{i,j}) \tag{3}$$
- $Pop : C \rightarrow 2^I$ associates to each class (leaf class or not) its own instances.

*Example 3.* Let's consider the class tree in Figure 5 where $A$ is a no-leaf class and $B$, $C$ and $D$ leaf-classes. We assume that each class has it own applicable properties, and the DBA (database administrator) has chosen its schema for $B$, $C$ and $D$ and a formula (2) or (3) for all non-leaf classes. To find the schema of the class $A$ using equation 2, we first perform the intersection operation among all properties of the schema of the leaf-classes. We obtain a set $U = \{b, c\}$, then we perform the intersection operation between $U$ and the applicable properties of $A$ ($\{a, b, c, g\}$). As result the schema of $A$ contains two properties $b$ and $c$ (see figure 5).

By using $Sch'$ definition (equation 3), $Sch'(A)$ would be $(a, b, c)$. The instances from $C$ and $D$ will be associated with NULL value for property $a$.

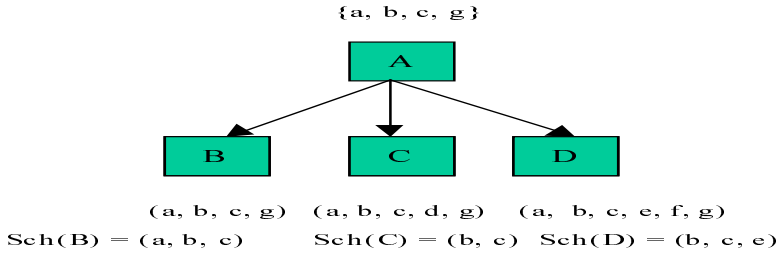


**Fig. 5.** An Example of a no-leaf Class Schema

## 6 Algorithms for integrating ontology-based database sources

In this section, we present algorithms to integrate various ontology-based database sources that correspond to the same domain. A typical scenario is the one of Web-services of a particular domain like traveling [18]. Each supplier references the same domain ontology and adds its own extensions. Let $S = \{SB_1, SB_2, ..., SB_n\}$ be the set of data sources participating in the data integration process. Each data source $SB_i$ $(1 \leq i \leq n)$ is defined as follows: $SBO_i =< O_i, I_i, Sch_i, Pop_i >$. We assume that all sources have been designed referencing as much possible a common shared ontology $O$. *As much possible* means that (1) each class of a local ontology references explicitly (or implicitly through its parent class its lowest subsumption class in the shared ontology and (2) only properties that do not exist in the shared ontology may be defined on a local ontology, otherwise it should be imported through the case-of relationship. This requirement is called smallest subsuming class reference requirement (S2CR2). Each source is designed following three steps:

1. The DBA of each source defines her own ontology $O_i :< C_i, P_i, Sub_i, Applic_i >$
2. The DBA of each source chooses for each leaf class properties that are associated with values by defining $Sch_i : C_i \rightarrow 2^{P_i}$;
3. The DBA choose an implementation of each leaf $c_i$ class (e.g., to ensure the third normal form), and then she defines $Sch(c_i)$ as a view over $c_i$ implementation.

We may distinguish two different integration approaches associated with automatic integration algorithms. These approaches are:

- *Fragmentation*: each local ontology of each class is a fragment of the shared ontology (Figure 2).
- *RealExtension*: each local ontology may be an extension of the shared ontology $O$ (to ensure the autonomy of a local source). This extension is done through explicit subsumption using case-of relationship (Figure 4) and should respect the S2CR2.

### 6.1 An Integration Algorithm for Fragmentation

[4] This integration approach assumes that the shared ontology is completed enough to cover the needs of all local sources. Such an assumption is done for instance in the Picsel2 project [18] for integrating web service (travel agency) or in COIN [7]. Source autonomy consists in selecting the pertinent subset of the shared ontology (classes and properties), and (2) designing the local database schema.

The ontology $O_i$ $(1 \leq i \leq n)$ of each source $SB_i$ 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:
$$Sch_{OInt}(c) = (\cap_{i \in \ \{1..n \ | Sch(cl) \neq \phi\}} Sch_i(c)) = \quad (4)$$
This definition ensures that instances of the integrated system are not expanded with null values to fit with the more precisely defined 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.
- The population of each class of the integrated system $Pop_{OInt}$ is defined as follows:
$$Pop_{OInt}(c) = \bigcup_i proj_{sch(c)} Pop_i(c) \quad (5)$$
where $proj$ is the projection operation as defined in classical databases.

### 6.2 An Integration Algorithm for RealExtension

In a number of cases including the target application domains of the PLIB approach, namely automatic integration of electronic catalogues/database of industrial components, more autonomy is requested by various sources:

- classification of each source needs to be completely different from the shared ontology;

[4] This approach corresponds to formula (2). An approach based on formula (3) is also possible

– some classes and properties do not exist at all in the shared ontology and need to be added in the local ontologies.

This case differs from the previous one by the fact that each data source has its own ontology and the classes of each ontology are specific (no class of the shared ontology are directly used in local ontology's). But all the ontologies reference "as much possible" (S2CR2) a shared ontology $O :< C, P, Sub, Applic >$.

Therefore, each source $SB_i$ 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 $SB_i$ is defined as quintuple: $SB_i =< O_i, I_i, Sch_i, Pop_i, M_i >$. In such as case also automatic integration is possible. To do so, 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 $< C^F, P^F, Sub^F, Applic^F >$, where element of these structures is defined as follows:

– Integrated classes $C^F = C \bigcup_{(i \mid 1 \leq i \leq n)} C_i$,
– $P^F = P \bigcup_{(i \mid 1 \leq i \leq n)} P_i$,
– $\forall c \in C,\ Sub^F(c) = Sub(c) \bigcup_{(i \mid 1 \leq i \leq n)} M_i(c)$
– $Applic^F(c) = \begin{cases} Applic(c), \text{ if } c \in C \\ Applic(c_i), \text{ if } c \in C_i \wedge c \notin C \end{cases}$
– Then, 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: $Pop^F(c) = Pop_i(c)$.

  Otherwise (i.e., $c$ belongs to the shared ontology tree), $Pop^F(c)$ is defined as follows:

$$Pop^F(c) = \bigcup_{(c_j \in Sub^F(c))} Pop^F(c_j) \tag{6}$$

– Finally, 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 $c$ does not belong to $C$ but to one $C_i$, $sch(c)$ is computed using the formula (2) (resp. 3).
  Otherwise(if $c$ belongs to the shared ontology), its schema is computed recursively using a poster-order tree search by :

$$Sch^F(c) = Applic(c) \bigcap \Big( \bigcap_{(c_j \mid c_j \ \in \ Sub^F(c) \wedge Pop^F \neq \phi)} Sch^F(c_j) \Big) \tag{7}$$

This shows that it is possible to leave a large autonomy to each local source and compute in a fully automatic, deterministic and exact way the corresponding integrated system. To the best of our knowledge our ontology based database approach is the first approach that reconciles these two requirements.

It is important to notice that when all data sources use independent ontologie without referencing a shared ontology the task of mapping these ontologies onto a receiver ontology may be done manually, by the DBA of the receiving system. But

then integration process will be performed automatically as in the RealExtension case.

## 7 Conclusion

In this paper, we present a new classification of integrated systems based on three critter's : (1) data representation, (2) the sense of the mapping between a global and local schemas, and (3) the nature of the mapping (manual, semi automatic and automatic). We also proposed a fully automated technique for integrating heterogeneous sources called ontology-based database integration approach. This approach assumes the existence of a shared ontology and guarantees the autonomy of each source by extending the shared ontology to define its local ontology. This extension is done by adding new concepts and properties. The ontologies used by our approach are modeled according to a formal, multilingual, extensible, and standardized (ISO 13584) known as PLIB. The fact that the ontology is embedded with each data source helps in capturing both the domain knowledge and the knowledge about data, schema, and properties. Therefore it allows a complete automation of the integration process contrary to the current existing techniques. Finally, two integration algorithms are presented: (1) when all sources only use a fragment of a shared ontology, and (2) when sources extend the shared ontology by specific classes and properties.

In addition to its capability for automating the integration process of heterogeneous databases (note that several prototypes of ontology-based databases are currently in progress in our laboratory), there are many other future directions that need to be explored. Some of the more pressing ones are: (1) extending our ontology model to capture functional dependencies between properties, (2) *schema evolution*, (3) considering the query optimization aspect to see how an ontology can be used for indexing query (semantic indexing) and (4) 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

1. S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber. Active xml: Peer-to-peer data and web services integration. *Proceedings of the International Conference on Very Large Databases*, pages 1087–1090, 2002.
2. L. Bellatreche, K. Karlapalem, and M. Mohania. 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, 2001.
3. S. Castano and V. Antonellis. Semantic dictionary design for database interoperability. *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 43–54, April 1997.

4. S. Castano, V. Antonellis, and S. D. C. Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, 2001.
5. S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous information sources. *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, pages 7–18, Marsh 1994.
6. F. Franois Goasdoué, V. Lattès, and M. C. Rousset. The use of carin language and algorithms for information integration: The picsel system. *International Journal of Cooperative Information Systems (IJCIS)*, 9(4):383–401, December 2000.
7. C.H. Goh, S. Bressan, E. Madnick, and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–293, 1999.
8. T. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1995.
9. N. Guarino and C. A. Welty. Ontological analysis of taxonomic relationships. *in Proceedings of 19th International Conference on Conceptual Modeling (ER'00)*, pages 210–224, October 2000.
10. F. Hakimpour and A. Geppert. Global schema generation using formal ontologies. *in Proceedings of 21th International Conference on Conceptual Modeling (ER'02)*, pages 307–321, October 2002.
11. R. Lawrence and K. Barker. Integrating relational database schemas using a standardized dictionary. *in Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 225–230, Marsh 2001.
12. A. Levy. Answering queries using views: a survey. *in the VLDB Journal*, 10(4):270–294, 2001.
13. A. Y. Levy, A. Rajaraman, and J. J. Ordille. 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)*, pages 43–55, June 1996.
14. E. Mena, V. Vipul Kashyap, A. Illarramendi, and A. P. Sheth. Managing multiple information sources through ontologies: Relationship between vocabulary heterogeneity and loss of information. *in Proceedings of Third Workshop on Knowledge Representation Meets Databases*, August 1996.
15. B. Omelayenko and D. Fensel. A two-layered integration approach for product information in b2b e-commerce. *Proceedings of the Second International Conference on Electronic Commerce and Web Technologies*, pages 226–239, September 2001.
16. G. Pierra. Context-explication in conceptual ontologies: The plib approach. *To appear in Proceedings of 10th ISPE International Conference on Concurrent Engineering: Research and Applications (ce'03) : Special Track on Data Integration in Engineering*, July 2003.
17. G. Pierra, J. C. Potier, and E. Sardet. From digital libraries to electronic catalogues for engineering and manufacturing. *International Journal of Computer Applications in Technology (IJCAT)*, 18:27–42, 2003.
18. C. Reynaud and G. Giraldo. An application of the mediator approach to services over the web. *Special track "Data Integration in Engineering, Concurrent Engineering (CE'2003) - the vision for the Future Generation in Research and Applications*, July 2003.
19. G. Terracina and D. Ursino. A uniform methodology for extracting type conflicts and & subscheme similarities from heterogeneous databases. *Information Systems*,

25(8):527–552, December 2000.
20. J. D. Ullman. Information integration using logical views. *Proceedings of the International Conference on Database Theory (ICDT), Lecture Notes in Computer Science*, 1186:19–40, January 1997.
21. P. R. S. Visser, M. Beer, T. Bench-Capon, B. M. Diaz, and M. J. R. Shave. Resolving ontological heterogeneity in the kraft project. *10th International Conference on Database and Expert Systems Applications (DEXA'99)*, pages 668–677, September 1999.
22. H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information - a survey of existing approaches. *Proceedings of the International Workshop on Ontologies and Information Sharing*, pages 108–117, August 2001.