

A Methodology for Building and Querying an Ontology representing Data and Multimedia Sources*

Domenico Beneventano
DII - Università di Modena e
Reggio Emilia
via Vignolese 905
Modena, Italia

domenico.beneventano@unimore.it

Claudio Gennaro
ISTI - CNR
via G. Moruzzi 1
Pisa, Italia

claudio.gennaro@isti.cnr.it

Francesco Guerra
DEA - Università di Modena e
Reggio Emilia
viale Berengario 51
Modena, Italia

francesco.guerra@unimore.it

ABSTRACT

Managing data and multimedia sources with a unique tool is a challenging issue. In this paper, the capabilities of the MOMIS integration system and the MILOS multimedia content management system are coupled, thus providing a methodology and a tool for building and querying a populated ontology representing data and multimedia sources.

1. INTRODUCTION

Semantic Web relies on ontologies to share a domain representation among different applications on the web. By providing a web application with the capabilities of reading and understanding ontologies, it is possible to automatically know which information is held by a web resource that refers to an ontology. By reconciling concepts belonging to different ontologies, it is possible to know without ambiguity the contents of web resources annotated according to different ontologies.

Nowadays, Semantic Web applications need to deal with populated ontologies, i.e. ontologies where it is possible to specify instances and assertions, for executing complex processes as semantic disambiguation, searching and application of advanced reasoning techniques.

Several ontologies are available (see, for example, the OpenCyc¹ knowledge base), but most of them are too general (or vice-versa too detailed) to be used in real general-purpose applications. Moreover, the process of populating an ontology with domain instances is a heavy issue, critical to be automatized: instances of a domain have to be identified and associated to the corresponding ontology concepts.

In [2], we proposed a method for building a populated domain ontology representative of a set of web data sources. The method exploits the capabilities of a mediator system

*The work presented in this paper has been partially supported by the Italian FIRB project RBNE05XYPW NeP4B - Networked Peers for Business.

¹<http://www.opencyc.org/>

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.

VLDB '08, August 24-30, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 000-0-00000-000-0/00/00.

to create an integrated view of a set of data sources, i.e. a domain ontology schema, and a set of annotations linking data to the integrated view, i.e. the ontology population.

This work is part of the NeP4B (Networked Peers for Business)² project, where we aim to contribute innovative ICTs solutions for SMEs, by developing an advanced technological infrastructure to enable companies of any nature, size and geographic location to search for partners, exchange data, negotiate and collaborate without limitations and constraints. In the NeP4B approach, data sources related to the same domain belong to the same semantic peer. Semantic peers are related by mappings, thus building a network.

In this paper we will focus on a single peer and we introduce a methodology for building and querying a Semantic Peer Data Ontology (SPDO), i.e. a populated ontology of “traditional” and “multimedia” data sources belonging to a peer. Such a work is based on the two previously developed MOMIS and the MILOS systems.

MOMIS (Mediator environment for Multiple Information Sources)³ [2] is a framework to perform integration of structured and semi-structured data sources, plus a query management environment able to process incoming queries on the integrated schema.

MILOS [1] is a Multimedia Content Management System tailored to support design and effective implementation of digital library applications. MILOS supports the storage and content based retrieval of any multimedia documents whose descriptions are provided by using arbitrary metadata models represented in XML.

Our approach aims at joining the methodologies implemented in both the systems: in particular, the MILOS methodology for representing, storing and indexing multimedia sources will allow the application of the MOMIS methodology to build and query an ontology representing data and multimedia sources.

2. THE SYSTEM AT A GLANCE

Our approach exploits and extends the MOMIS and MILOS methodologies, where the MOMIS methodology is exploited for building and querying the SPDO, MILOS is exploited for managing the interactions with the multimedia sources. As depicted in Figure 1, MOMIS relies on wrappers for interacting with the sources. A wrapper has to

²<http://www.dbgroup.unimo.it/nep4b>

³<http://www.dbgroup.unimo.it/Momis> for publications about MOMIS

accomplish two tasks: 1. translating the descriptions of the sources into a common language (ODL_{J3}) representation. In such a way, the MOMIS methodology may be applied to the sources independently of their structures and contents. The translation from the typical languages used for representing data sources and ODL_{J3} is quite straightforward. In the case of multimedia sources, special functionalities, provided by MILOS, are required for representing the specificities of this kind of sources.

2. executing query at the local source level. The MOMIS Query Manager is able to translate a query on the SPDO (a *global query*) into a set of local queries to be locally executed by means of wrappers.

An interface allows users to browse and query the SPDO. Querying the SPDO means to identified instances in the populated ontology satisfying the constraints established by a user via a graphical interface or directly an SQL query.

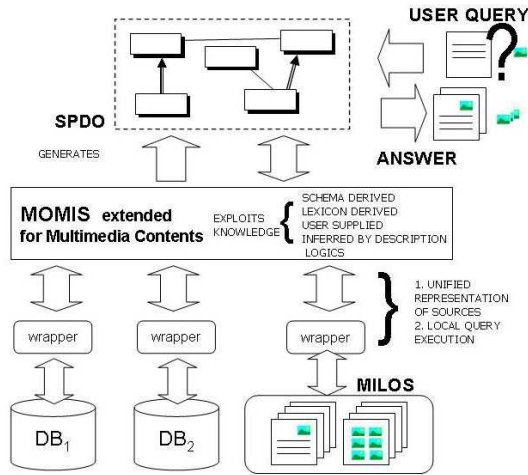


Figure 1: The functional architecture

3. A UNIFIED VIEW OF DATA AND MULTIMEDIA SOURCES

Data sources are described by means of the ODL_{J3} object oriented language. ODL_{J3} is a source independent language used for describing heterogeneous schemas of structured and semistructured data sources in a common way. ODL_{J3} is very close to the standard ODL language⁴, that extends with the following operators: (1) Union constructor introduced to express alternative data structures in the definition of an ODL_{J3} class, thus capturing requirements of semistructured data; (2) Optional constructor introduced for class attributes to specify that an attribute is optional for an instance.

3.1 Representing multimedia within the SPDO

Multimedia sources (MMS) are managed and queried by means of the Content Management System MILOS specialized to support multimedia documents.

Each MMS is described through a schema expressed in ODL_{J3} called Local Multimedia Schema (LMS) that encapsulates specific predefined data-types for the management of multimedia data. In particular, in LMS there is a set of attributes, declared using standard predefined ODL_{J3}

⁴<http://www.odmg.org>

types (such as string, double, integer, etc.), along with a set of multimedia metadata declared by means of two special predefined classes, which support the similarity operator \sim , namely **Text** and **Image**. **Text** class encapsulates natural language descriptions searchable through the full-text search paradigm. **Image** class encapsulates visual descriptors encoded in MPEG-7 extracted from data objects (photos, videos, etc). An instance of a **Image** corresponds to an instance of a multimedia data (e.g., an image), which is uniquely identified by an *URI*.

Thus, an LMS comprises: a variable part done by one or more traditional attributes, an optional **Image** attribute and an optional **Text** attribute.

3.2 Representing the SPDO

We build a conceptualization of a set of data and multimedia sources, composed of global classes and global attributes and mappings between the SPDO and the local schemata. We follow a *GAV* approach, thus this mapping is expressed by defining, for each global class G , a mapping query q_G over the local schemata. This mapping query is defined in a semi-automatic way as follows:

1. A *Mapping Table* (MT) is specified for each global class G , whose columns represent the local classes $L(G)$ belonging to G and whose rows represent the global attributes of G . An element $MT[G][L]$ represents the set of local attributes of L which are mapped onto the global attribute GA .

2. *Data Conversion Functions* are manually specified and, for each not null element $MT[G][L]$, establish the operations that transform the values of the local attributes of L into the values of the corresponding the global attribute GA .

3. *Join Conditions* are defined between pairs of local classes belonging to G and allow the system to identify instances of the same real-world object in different sources. Automatic object identification techniques (see for example [5]) or the designer knowledge may be exploited to define correct join conditions.

4. *Resolution Functions* [6, 3] are introduced for global attributes to solve data conflicts of local attribute values associated to the same real-world object. In [3] several resolution are described and classified; in our framework we consider, in particular, the *PREFERRED* function, which takes the value of a preferred source and the *RANDOM* function, which takes a random value. If the designer knows that there are no data conflicts for a global attribute mapped onto more than one source (that is, the instances of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an *Homogeneous Attribute*; for homogeneous attributes resolution functions are not necessary. By using the introduced concepts, the mapping query is defined on the basis of the *full outerjoin-merge* operator introduced in [6]: for a global class G , the mapping query q_G is obtained by performing the *full outerjoin-merging* of the local classes $L(G)$ belonging to G .

Let us consider, for example, a scenario related to the tourist domain. In this context, we can observe many portals and websites promoting different data. However, the information about a tourist service, location or event is frequently widespread in different specific websites, due to the specialization of the information publisher. Thus, if a user wants to have complete knowledge about a location, s/he has to navigate throughout several websites. This issue gener-

ates multiple researches on search engines, where it is easy to find both incomplete and overlapping information. An application, which provides a unified view of the data provided by different web sources, may provide the tourist promoters and travelers all the information about a location by means of only one tool.

According to the depicted scenario, let us consider a global class *Hotel* that integrates data from two local classes *resort* and *hotel*. The mapping table of such Global Class takes into account the new data types **Text** and **Image** introduced by multimedia sources. In particular, only “homogenous” mappings: **Text-to-Text**, **Image-to-Image**, are allowed.

Hotel	resort	hotel
name (join)	Name	denomination
telephone	Telephone	tel
fax	Fax	fax
www	Web-site	www
room_num	Room_num	rooms
price (RF)	Price_avg	mean_price
city	City	location
stars	Stars	-
free_wifi	-	free_wifi
photo	Photo	img
description	Description	commentary

According to the mappings, we specify the *name* as join attribute intending that instances of the classes *resort* and *hotel* having the same name represent the same real object. Moreover, a resolution function may be defined for the global attribute *price*, i.e. the value of the global attribute *price* is the average of the values assumed by both the local sources.

4. QUERYING STRUCTURED AND MULTIMEDIA DATA

We consider a scenario where:

Data sources support queries with a $\langle \text{data_cond} \rangle$ expressed with set oriented operators typical of structured and semi-structured data, such as =, <, >, ...

Multimedia sources support, besides a $\langle \text{data_cond} \rangle$ on “standard” attribute, a $\langle \text{multimedia_cond} \rangle$, which exploits the similarity operator \sim and expresses both full-text search on textual attributes (declared as **Text** type) and similarity search on MPEG-7 attributes (declared as **Image** type). The \sim operator introduces the problem of complex query search, when more similarity conditions are combined using the AND operator. To determine the top *k* hotels, that is, *k* objects with the highest overall scores, the naive algorithm must access every object in the database, to find its score under each attribute.

The *MIN* operator is used as fuzzy AND to combine the scores (ranging from 0 to 1) of two streams of results sets retrieved by the full-text part of the query and visual similarity part. In [4], Fagin addresses this problem for a user query involving several multimedia attributes.

4.1 Querying the SPDO

Given a global class *G* with either multimedia attributes, denoted by $\langle \text{multimedia_attr} \rangle$ (as *photo* and *description* in the class *Hotel*) and “standard” attributes, denoted by $\langle \text{data_attr} \rangle$ (as *name* and *city* in the class *Hotel*), a query on *G* (*global query*) is a conjunctive query, expressed in a

simple abstract SQL-LIKE syntax as:

$$Q = \begin{aligned} &SELECT \langle \text{data_attr} \rangle, \langle \text{multimedia_attr} \rangle \\ &FROM G \\ &WHERE \langle \text{data_cond} \rangle \\ &AND \langle \text{multimedia_cond} \rangle \end{aligned}$$

where $\langle \text{data_cond} \rangle$ is on “standard” attributes of *G* and $\langle \text{multimedia_cond} \rangle$ is expressed, with the operator similarity operator \sim , on multimedia attributes of *G*.

To answer a global query on *G*, the query must be rewritten as an equivalent set of queries (*local queries*) expressed on the local classes *L(G)* belonging to *G*. This query rewriting is performed by considering the mapping between the SPDO and the local schemata; in a GAV approach the query rewriting is performed by means of **query unfolding**, i.e., by expanding the global query on *G* according to the definition of its mapping query q_G .

In this section we give an intuitive and informal description of the query unfolding process for $\langle \text{data_cond} \rangle$ and later on we show how to extend this process to $\langle \text{multimedia_cond} \rangle$.

4.1.1 Query unfolding for $\langle \text{data_cond} \rangle$

Let us consider the following global query:

```
select Name,www from Hotel
where price<100 and stars=3 and free_wifi=TRUE
```

The process for executing the global query consists of the following steps:

1. Computation of Local Query conditions: constraints on the global query are rewritten into corresponding constraints supported by the local classes. For example, the constraints **stars = 3** is translated into a constrain **Stars = 3** considering the local class **resort** and is not translated into any constraint considering the local class **hotel**.

2. Computation of Residual Conditions: Conditions on not homogeneous attributes cannot be translated into local conditions: they are considered as *residual* and have to be solved at global level. As an example, let us suppose that a numerical global attribute (as for instance **price** in our example) GA is mapped onto *L1* and *L2*, and an AVG function is defined as resolution function, the constraint (GA = value) cannot be pushed at the local sources, since the AVG function has to be calculated at a global level and the constraint may be globally true but locally false. It is easy to verify that the same happens defining for **price** the *PREFERRED* or the *RANDOM* resolution function.

3. Generation and execution of local queries: for each local source involved in the global query, a local query is generated. The select list is obtained with the union of the attributes of the global select list, of the Join Condition and of the Residual Condition; these attributes are transformed into the corresponding ones at the local level on the basis of the Mapping Table.

```
LQ_resort = select Name,Price from resort
where Stars = 3
```

```
LQ_hotel = select denomination,mean_price
from hotel where free_wifi= TRUE
```

The queries are executed on the local sources.

4. Fusion of local answers : The local answers are fused into the global answer on the basis of the mapping

query q_G defined for G , i.e. by using a Full Outerjoin-merge operation. Intuitively, this operation is substantially performed in two steps:

1. Computation of the **full join** of local answers (FOJ):

```
LQ_resort full join LQ_hotel on
(LQ_resort.Name=LQ_hotel.Denomination)
```

2. **Application of the Resolution Functions:** for each attribute GA of the global query the related Resolution Function is applied to FOJ thus obtaining a relation R_{FOJ} ; in our example the result is the relation $R_{FOJ}(name, www, price)$.

5. Application of the Residual Condition: the result of the global query is obtained by applying the residual condition to the R_{FOJ} :

```
select name,www from R_FOJ where price < 100
```

4.1.2 Query unfolding for $\langle multimedia_cond \rangle$

Our system implements two kinds of multimedia query. First, global query expressing one or more $\langle multimedia_attr \rangle$ in the select list, without the expression of any $\langle multimedia_cond \rangle$; for example, referring to the Mapping Table described in Section 3.2, if **photo** is in the select list, a resolution function has to be applied for selecting which photo must be returned to the user in the case of hotels having the same name and satisfying the $\langle data_cond \rangle$ in both the sources. Since no $\langle multimedia_cond \rangle$ has been expressed, the system returns the photo of both the local sources.

Second, global query containing one or more $\langle multimedia_cond \rangle$, that is solved by applying the procedure described for the data sources and the resolution functions defined by the user for the multimedia conditions. With reference to the example of section 3.2, let us introduce a simple $\langle multimedia_cond \rangle$, i.e. involving one multimedia attribute, while the select list may contain more than one multimedia attributes:

```
select name, www, photo, description
from Hotel
where price < 100 and
      stars = 3 and free_wifi= TRUE and
      photo ~ "http://www.hotels.com/imgs/x123.jpg"
```

The obtained local queries are:

```
LQ_resort =
select Name, Web-site, Photo, Description
from resort where Stars = 3 and
Photo ~ "http://www.hotels.com/imgs/x123.jpg"
```

```
LQ_hotel =
select denomination, www, img, commentary
from hotel where free_wifi= TRUE and
img ~ "http://www.hotels.com/imgs/x123.jpg"
```

Intuitively, each local query produces two result lists, where each item is ordered according to the similarity of the local **Image** attributes with the given sample image.

Since *name* attribute as been defined as join attribute, the integration of the two result lists coming from the MMSs, is a simple task: all the records having the same *name* attribute (i.e., $Name=denomination$) are fused. After the FOJ computation, the resolution function defines which multimedia

object (the one represented by the attribute **photo** in the local source resort or the attribute **img** of the source hotel) returns to the user. For instance, the one with greater similarity score.

As a more general case, let us consider a natural extension of the previous query, expressing more than one $\langle multimedia_cond \rangle$:

```
select Name, www, photo, description
from Hotel
where price < 100 and stars = 3 and
      free_wifi= TRUE and
      photo ~ "http://www.hotels.com/imgs/x123.jpg"
      and description ~ "private beach"
```

At the local level, MILOS will process each sub-query independently producing a result list with combined (Image/Text) scores. Therefore, the application of the resolution function establishes if either returning all pairs of multimedia objects or only pairs with greater similarity score.

```
LQ_resort =
select Name, Web-site, Photo, Description
from resort where Stars = 3 and
Photo ~ "http://www.hotels.com/imgs/x123.jpg"
and Description ~ "private beach"
```

```
LQ_hotel =
select denomination, www, img, commentary
from hotel where free_wifi= TRUE and
img ~ "http://www.hotels.com/imgs/x123.jpg"
and commentary ~ "private beach"
```

5. CONCLUSION

We presented a methodology implemented in a tool that allows a user to create and query a populated ontology representing data and multimedia sources. The methodology joins and extends the capabilities of two previously developed tools: the MOMIS integration system and the MILOS multimedia content management system.

6. REFERENCES

- [1] G. Amato, C. Gennaro, P. Savino, and F. Rabitti. Milos: a Multimedia Content Management System for Digital Library Applications. In *Proceedings of ECDL*, volume 3232 of *LNCS*, pages 14–25. Springer, September 2004.
- [2] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing*, 7(5):42–51, 2003.
- [3] J. Bleiholder and F. Naumann. Conflict handling strategies in an integrated information system. In *Proceedings of the WWW Workshop IIWeb*, 2006.
- [4] R. Fagin. Fuzzy queries in multimedia database systems. In *Proceedings of PODS*, pages 1–10, New York, NY, USA, 1998. ACM.
- [5] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Eng. Bull.*, 29(2):21–31, 2006.
- [6] F. Naumann, J. C. Freytag, and U. Leser. Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615, 2004.