

# A Mediator System for Data and Multimedia Sources <sup>\*</sup>

Domenico Beneventano<sup>1</sup> and Sonia Bergamaschi<sup>1</sup> and Claudio Gennaro<sup>2</sup> and  
Francesco Guerra<sup>3</sup> and Matteo Mordacchini<sup>2</sup> and Antonio Sala<sup>1</sup>

<sup>1</sup> DII - Università di Modena e Reggio Emilia  
Via Vignolese 905, 41100 Modena- Italy  
`firstname.lastname@unimore.it`

<sup>2</sup> ISTI - CNR,  
Via Moruzzi 1, Pisa - Italy  
`firstname.lastname@isti.cnr.it`

<sup>3</sup> DEA - Università of Modena and Reggio Emilia,  
Viale Berengario 51, 41100 Modena- Italy  
`firstname.lastname@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 an integrated virtual view of data and multimedia sources.

## 1 Introduction

The research community has been developing techniques for integrating heterogeneous data sources for the last twenty years. One of the main motivations is that data often reside in different, heterogeneous and distributed data sources that users need to access in a single and unified way to gather a complete view of a specific domain. A common approach for integrating information sources is to build a mediated schema as a synthesis of them. By managing all the collected data in a common way, a global schema allows the user to pose a query according to a global perception of the handled information. A query over the mediated schema is translated into a set of sub-queries for the involved sources by means of automatic unfolding-rewriting operations taking into account the mediated and the sources schemata. Results from sub-queries are finally unified by data reconciliation techniques.

Integration systems relying on mediator-based architectures have been developed, and most of them are able to integrate at the same time heterogeneous data sources, i.e. sources that represent their contents with relational, object-oriented and semi-structured (XML and its derived standards) models.

---

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

Similarity search for content-based retrieval (where content can be any combination of text, image, audio/video, etc.) has gained importance in recent years, also because of the advantage of ranking the retrieved results according to their proximity to a query. The systems usually exploit data types such as sets, strings, vectors, or complex structures that can be exemplified by XML documents. Intuitively, the problem is to find similar objects with respect to a query object according to a domain specific distance measure. However, the growing need to deal with large, possibly distributed, archives requires specialized indexing support to speedup retrieval. The common assumption is that the costs to build and to maintain an index structure are lower compared to the ones that are needed to execute a query.

In this paper we present an approach that extends the action sphere of traditional mediator systems allowing them to manage “traditional” and “multimedia” data sources at the same time. The result is implemented in a tool for integrating traditional and multimedia data sources in a virtual global schema and transparently querying the global schema . We believe this is an interesting achievement for several reasons.

Firstly, the application domain: there are several use cases where joining traditional and multimedia data is relevant (see Section 2 for a concrete scenario).

Secondly, multimedia and traditional data sources are usually represented with different models. While there is a rich literature for transforming the differently modelled traditional data sources into a common model and it is possible to represent different multimedia sources with a uniform standard model such as MPEG-7, a standard for representing traditional and multimedia data does not exist.

Finally, different languages and different interfaces for querying “traditional” and “multimedia” data sources have been developed. The former relies on expressive languages allowing expressing selection clauses, the latter typically implements similarity search techniques for retrieving multimedia documents similar to the one provided by the user.

This work is part of the NeP4B (Networked Peers for Business)<sup>4</sup> project, where we aim to contribute innovative Information and Communication Technology (ICT) solutions for Small and Medium Enterprises (SME), 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 project, we assume that data sources related to the same domain belong to the same semantic peer, i.e. a super peer exposing a semantic layer. Semantic peers are related by mappings, thus building a network.

In this paper we will focus on a single semantic peer and we will introduce a methodology and a system for building and querying a Semantic Peer Data Ontology (SPDO), i.e. a global schema including traditional and multimedia data sources related to a domain. The developed system is based on the MOMIS and the MILOS systems.

---

<sup>4</sup> <http://www.dbgroup.unimo.it/nep4b>

MOMIS (Mediator envirOnment for Multiple Information Sources)<sup>5</sup> is a framework to perform integration of structured and semi-structured data sources, with a query management environment able to process incoming queries on the integrated schema [2]. The MOMIS integration process gives rise to a Global Virtual View (GVV), in the form of global classes and global attributes, which may be considered as a domain ontology for the data sources.

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 work extends the MOMIS integration capabilities in two directions: language extension to capture the semantics of multimedia data types and a multimedia wrapper to automatically extract multimedia data source schemata. Moreover, it discusses problems and solutions to extend a mediation query processing to deal with multimedia data.

The paper is organized as follows: Section 2 describes an applicative scenario. Section 3 proposes an overview of the system; Section 4 extends the mediation methodology for building and integrating the SPDO of “traditional” and “multimedia” data sources. Section 5 introduces the problem of querying structured and multimedia data. Finally, in Section 6, we sketch out some conclusion and future work.

## 2 An applicative scenario

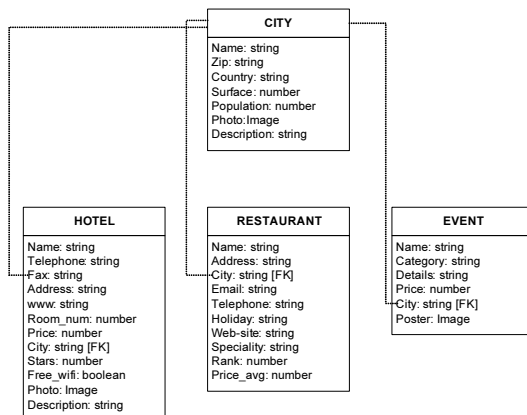
Let us consider the tourist domain where we can find many portals and websites. The information about a tourist service, location or event is frequently widespread in different specific websites, due to the specialization of the information publisher. Thus, if a user wants to have a complete knowledge about a location, s/he has to navigate through several websites. This issue generates multiple queries with search engines, retrieving 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 a more complete and easy to find information. Moreover, since our approach provides only a unified representation of data which still reside on specialized websites, the possibility of having out-of-date information is the same as from navigating the single specialized websites one at a time.

In Figure 1 we provide a graphical representation of the schema obtained from the integration of three data sources. It is composed of 4 classes: *hotel*, *restaurant*, *event* which are related to each other by means of the *city* class. Special attributes of the City, Hotel and Event classes are *Photo*, and *Poster* which are Multimedia Objects. These Multimedia Objects will be annotated

---

<sup>5</sup> See <http://www.dbgroup.unimo.it/Momis> for more publications about MOMIS



**Fig. 1.** The tourist integrated schema

with the MPEG-7 standard. Each data source contains traditional data types (e.g. city name in the Event class) together with multimedia data type (e.g. poster in the same Event class). Current search engines for multimedia content perform retrieval of similar objects using advanced similarity search techniques. The NeP4B project aims at combining the exact search on traditional data with a similarity search on multimedia data, exploiting the SPDO obtained by integrating the different data sources (traditional as well as multimedia). As an example, consider a user who wants to attend an event which involves fireworks. Using advanced search techniques over multimedia documents, in this case “poster”, it will be possible to search for posters that contain “festival” and related images of fireworks.

### 3 The system at a glance

Our approach exploits and extends the MOMIS and MILOS systems, where MOMIS is exploited for building and querying the SPDO, MILOS for managing the interaction with the multimedia sources.

As depicted in Figure 2, MOMIS relies on wrappers for interacting with the sources. A wrapper has to accomplish two tasks:

1. translating the descriptions of the sources into a common language ( $ODL_{I^3}$ ) representation. The translation from the usual modelling languages (relational, object, XML, OWL) and  $ODL_{I^3}$  is quite straightforward. In the case of multimedia sources, special functionalities, provided by MILOS, are required for representing the multimedia data types (see Section 4).
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. As shown in Section 5, MILOS will manage the execution of queries on the multimedia sources.

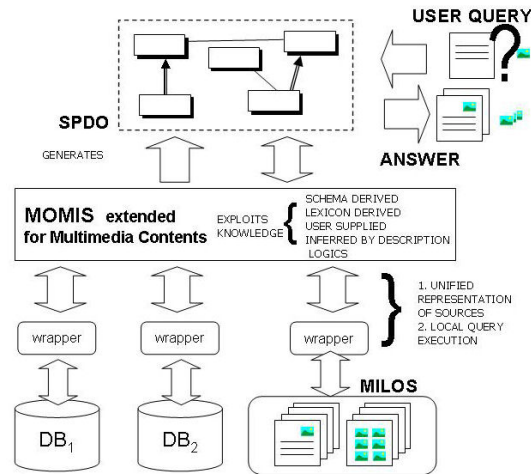


Fig. 2. The functional architecture

The integration of the MOMIS and the MILOS systems is achieved by means of the Web Services technology. The RMI of MILOS allows MOMIS to search by invoking the method:

```
search(String LMSClassName, String[] values,
       String[] fields, String[] operators, String returnField)
```

where the `fields` parameter is an array of (application known) names of multimedia attributes of the LMS (Local Multimedia Sources ClassName) to search for. The `values` parameter specifies the values that the fields must match (the different fields are searched by using the connective *AND*). The `operators` parameter specifies the matching operator to be used (i.e.: the standard XQuery operators “=”, “>”, “<”, “>=”, “<=”, “!=” and the similarity operator “~” for similarity search on multimedia attributes as shown in next section). Finally, `returnField` specifies the field of the retrieved records that the application wants to know. When a user submits a query on the global schema, MOMIS takes the query and produces a set of translated sub-queries to be sent to each involved data source. If the local source is a multimedia one, MOMIS translates the local query into an invocation of the `search` method following the technique described in section 5.2.

#### 4 A unified view of data and multimedia sources

The main features of the  $ODL_{J3}$  language, used for representing the local sources and the SPDO and the main issues related to the management of multimedia sources with MILOS are described.

$ODL_{J3}$  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 language<sup>6</sup>, that extends with the following operators: (1) Union constructor introduced to express alternative data structures in the definition of an ODL<sub>J3</sub> class, thus capturing requirements of semistructured data. (2)Optional constructor introduced for class attributes to specify that an attribute is optional for an instance.

#### 4.1 Representing multimedia data within the SPDO

Multimedia sources (MMS) are managed and queried by means of MILOS.

Each MMS is described through an ODL<sub>J3</sub> schema (although it is internally implemented in XML) 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  $n$  of attributes  $a_1, \dots, a_n$ , declared using standard predefined ODL<sub>J3</sub> types (such as string, double, integer, etc.). These attributes are referred to as *standard attributes* and support set oriented operators typical of structured and semi-structured data, such as =, <, >, ... Along with the standard attributes, LMS includes a set of  $m$  attributes  $s_1, \dots, s_m$ , declared by means of special predefined classes, which support the similarity operator  $\sim$ . The query on these attributes returns a ranked list sorted by descending relevance. We refer to these attributes as *multimedia attributes*. For instance, we can have a **Text** class that encapsulates natural language descriptions searchable through the full-text search. An **Image** class that encapsulates visual descriptors encoded in MPEG-7 extracted from data objects (photos, videos, etc).

The following example shows the declaration of the city class, which comprises five standard attributes and three multimedia attributes.

```
interface city() {
    // standard attributes
    attribute string Name;
    attribute string Zip;
    attribute string Country;
    attribute integer Surface;
    attribute integer Population;

    // multimedia attributes
    attribute Image Photo;
    attribute Text Description;
}
```

#### 4.2 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 these mappings are expressed by defining, for each global class  $G$ , a mapping query  $q_G$  over the local schemata.

---

<sup>6</sup> <http://www.odmg.org>

This mapping query is defined in a semi-automatic way (i.e. the designer is supported by the system to define the mapping query) 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  $MT[GA][L]$  represents the set of local attributes of  $L$  which are mapped onto the global attribute  $GA$ . The Mapping Table is automatically generated during the integration process, as described in [2].
2. *Data Conversion Functions* are manually specified and, for each not null element  $MT[GA][L]$ , establish the operations that transform the values of the local attributes of  $L$  into the values of the corresponding 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 [7]) or the designer knowledge may be exploited to define correct join conditions.
4. *Resolution Functions* [8, 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 functions are described and classified; in our framework we consider and implement some of such resolution functions, 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), s/he can define this attribute as an *Homogeneous Attribute*; of course, for homogeneous attributes resolution functions are not necessary. A global attribute mapped into only one local class is a particular case of an homogeneous attribute.

By using the introduced concepts, the mapping query is defined on the basis of the *full outerjoin-merge* operator introduced in [8]: 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 the ODL<sub>J3</sub> description of the global class *Hotel* of Figure 1. This class is the result of the integration of the two local classes *resort* and *hotel*, having description:

```
interface resort() {
    attribute string    Name;
    attribute string    Telephone;
    attribute string    Fax;
    attribute string    Web-site;
    attribute integer   Room_num;
    attribute integer   Price_avg;
    attribute string    City;
```

```

    attribute integer Stars;
    attribute Image Photo;
    attribute Text Description;
}

interface hotel() {
    attribute string denomination;
    attribute string tel;
    attribute string fax;
    attribute string www;
    attribute integer rooms;
    attribute integer mean_price;
    attribute string location;
    attribute boolean free_wifi;
    attribute integer stars;
    attribute Image img;
    attribute Text commentary;
}

```

The mapping table of such Global Class takes into account the multimedia attributes `Commentary`, `Description` (Text) and `Photo`, `img` (Image) introduced by multimedia sources. In general, only “homogenous” mappings between multimedia attributes are permitted, e.g. `Text-to-Text`, `Image-to-Image`, etc.

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* (*denomination*) 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 maximum of the values assumed by both the local sources.

## 5 Querying structured and multimedia data

According to the functional architecture depicted in figure 2, the proposed approach relies on the engines of the local sources for the execution of the queries.

Thus, the query processing has to manage the heterogeneity that such architecture entails which is disclosed in different operator supported by different local sources. We consider a scenario where:

1. *Traditional data sources* support queries with set oriented operators typical of structured and semi-structured data, such as =, <, >, ... More precisely the condition of such kinds of query, denoted by <data\_cond>, is a conjunction of positive atomic constraints (*attribute op value*) or (*attribute<sub>1</sub> op attribute<sub>2</sub>*) with *op* a relational operator. Our proposal exploits generic DBMSs for managing data sources.
2. *Multimedia data sources* support, besides a <data\_cond> on “standard” attribute, a <multimedia\_cond> which express full-text search on textual attributes (declared as **Text** type), similarity search on MPEG-7 attributes (declared as **Image** type). We consider multimedia sources managed by the MILOS system, as explored in the next Section.

### 5.1 Processing Queries on Multimedia Sources

Let us consider the multimedia class *city* described in section 4.1, and suppose that a user would like to find all the cities that have zip code equal to 41100, image similar to the one given as example (by providing its URL), and that also best match the given textual description. The following query expresses, in an SQL-like query language, the user request:

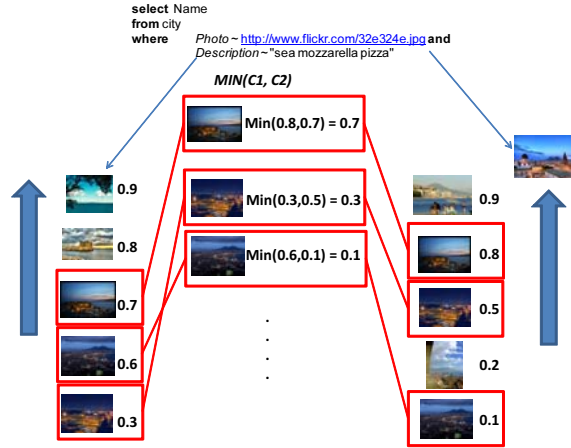
```
select Name from city
where zip = '41100' and photo ~ "http://turismo.comune.modena.it/38.jpg"
      and description ~ "cathedral"
```

The MILOS system is responsible for the query execution. A particular attention has been paid to the implementation of the  $\sim$  operator, in case of nearest neighbors queries (as it is in our case). This operator, in fact, introduces the problem of complex query search, when more similarity conditions are combined using the AND operator. To determine the top  $k$  cities, 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.

In Figure 3 we show an example of query on multimedia sources, where 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 the visual similarity part. In [5], Fagin addressed this problem for a user query involving several multimedia attributes. Notice that MILOS generates a unique combined score even if more than one similarity condition is expressed.

### 5.2 Querying the SPDO

Given a global class  $G$  with either multimedia attributes, denoted by <multimedia\_attr> (such as **photo** and **description** in the class **Hotel**) and “standard” data type attributes, denoted by <data\_attr> (such as **name** and **city** in the class **Hotel**),



**Fig. 3.** Example of multimedia query processing.

a query on  $G$  (*global query*) is a conjunctive query, expressed in a simple abstract SQL-like syntax as:

$$\begin{aligned}
 Q &= \text{SELECT } \langle \text{data\_attr} \rangle, \langle \text{multimedia\_attr} \rangle \\
 &\text{FROM } G \\
 &\text{WHERE } \langle \text{data\_cond} \rangle \\
 &\text{AND } \langle \text{multimedia\_cond} \rangle
 \end{aligned}$$

where  $\langle \text{data\_cond} \rangle$  is on standard data type attributes of  $G$  and  $\langle \text{multimedia\_cond} \rangle$  is expressed, with the 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 the case of “standard” local data sources, the local execution of a query produces a list of objects matching  $\langle \text{data\_cond} \rangle$ . In the case of local multimedia data repositories, the  $\sim$  operator is used to ideally re-order all the objects on the basis of their similarity expressed by the  $\langle \text{multimedia\_cond} \rangle$ . The final result will be a ranked list of the objects of each multimedia source. At the global level, the system needs to combine all this local results. We choose to fuse the results by using a Full Outerjoin-merge operation between the lists. Multimedia result lists are merged by means of both ranks and full join conditions in order to obtain a unique multimedia ranked list of results. The join conditions are defined by means of a join attribute, which is a “standard” attribute present in

all local classes. This list has to be further fused with the results coming from “standard” sources.

In this section we give an intuitive and informal description of all these processes, describing the query unfolding process for `<data_cond>` and showing how to extend this process to `<multimedia_cond>`.

**Query unfolding for `<data_cond>`** Let us consider the following global query:

```
select Name, www, price 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. In particular, an atomic constraint (*GA op value*) is translated into a constraint on the local class *L* as follows:

```
(MTF[GA][L] op value)
  if GA is an homogeneous attribute and
  MTF[GA][L] is not null and
  the op operator is supported into L
true otherwise
```

where  $MTF[GA][L]$  is the data conversion function defined for the local attribute *L* with respect to the global attribute *GA*. For example, the constraint `stars = 3` is translated into a constraint `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 the 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 `MAX` 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_avg, Web-site from resort
where Stars = 3
```

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

These 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:

- (a) Computation of the **full join** of local answers (FOJ):

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

- (b) **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  $R_{FOJ}(name, www, price)$ . Notice that if the  $www$  attribute does not have any Resolution Function applied, i.e. we hold all the values, all the  $www$  values for a certain hotel  $X$  are retained: we will have only one record for  $X$  in  $R_{FOJ}$ , where the value of the  $www$  attribute is the concatenation of the  $www$  values (obviously adequately separated, e.g. by a comma).

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, price from R_FOJ where price < 100
```

**Query unfolding for  $\langle \text{multimedia\_cond} \rangle$**  Let us start our discussion about  $\langle \text{multimedia\_cond} \rangle$  with the simplest case where multimedia attributes are mapped on a single local source: for the global class `Hotel` we consider  $M[\text{resort}][\text{Photo}] = \text{NULL}$  and  $M[\text{hotel}][\text{description}] = \text{NULL}$ . Let us consider a query with a  $\langle \text{multimedia\_cond} \rangle$ :

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

By applying the unfolding process described before, there is no residual condition since all the involved attributes are homogeneous; the local queries are:

```
LQ_resort = select Name, Description from resort
           where Stars= 3 and
           and Description ~ "private beach"
```

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

Each local query produces a stream of tuples in order of decreasing similarity w.r.t. the given query. In the fusion step, tuples of *LQ\_resort* and *LQ\_hotel* are joined on the condition (*LQ\_resort.Name=LQ\_hotel.Denomination*) and the resulting FOJ provides the global query answer since it is necessary to apply neither resolution functions nor residual conditions.

Obviously, in general, there is no need to compute the whole combined result list. Rather, the query evaluation typically retrieve a ranked result set, where an aggregated score (from *LQ\_resort* and *LQ\_hotel*) is attached to each tuple returned. In fact, only a few top-ranked results are normally of interest to the user. This is exactly the same complex query problem that arises in processing queries on local multimedia sources discussed in Section 5.1. However, although the same solution applies also in this case, we adopt a more general approach, applicable to any situation, as shown in the following.

Let us consider now the general case where a multimedia attribute is mapped on more than one local source. As discussed before, (1) we need to define a resolution function for such attribute and (2) constraints on such attribute cannot be fully solved on local sources and need to be solved at global level.

To describe this problem, let us suppose that the global attribute *photo* is mapped on both the local classes *resort* and *hotel* and let us consider a query with a *<multimedia\_cond>* on such a global attribute *photo*:

```
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"
```

If a *PREFERRED* or a *RANDOM* resolution function is defined for the attribute *photo*, according to the query unfolding process described in the previous section, the constraint for the attribute *photo* needs to be solved at the global level and then it will not be rewritten at the local level.

On the other hand, in the case of multimedia attributes, besides the “normal” resolution functions, such as *PREFERRED* or *RANDOM*, more appropriate resolution functions may be considered. For this reason, we defined a new resolution function, called *MOST\_SIMILAR*, which returns the multimedia objects most similar to the one expressed in the *<multimedia\_cond>*.

In the following we discuss the query unfolding process in the presence of this resolution function; note that only the query unfolding steps concerning the constraint on *photo* are described.

In step 3 the constraint on *photo* is rewritten at the local level as:

```
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 a result list where each “local record” has a score and is ordered according to the similarity of the local *Image* attributes with the given sample image.

In step 4, after the FOJ computation, the *MOST\_SIMILAR* function must be “applied”: for records with a not null “multimedia object” both for *img* and *Photo*, we need to identify the “multimedia object” most similar to the one expressed in the  $\langle \text{multimedia\_cond} \rangle$ . Since the local query results are *congruent*, i.e. are related to the same  $\langle \text{multimedia\_cond} \rangle$ , the *MOST\_SIMILAR* function exploits the associated scores for selecting the multimedia results.

In the case of *not congruent* results, as in the following example, another technique has to be applied. Suppose that *hotel* contains only the multimedia attribute *img*, and *resort* contains both the multimedia attribute *Photo* and *Description*. Consider now a query with a  $\langle \text{multimedia\_cond} \rangle$  on both such global 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"
      and description ~ "private beach"
```

In step 3 multimedia constraints are rewritten at the local level as:

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

The local execution of both queries generates a list of results, each one with a score. However, the scores are “not congruent”, since the one coming from *LQ\_resort* is the product of a combination of two scores (with the fuzzy **AND** for instance), while the list coming from *LQ\_hotel* has scores that correspond only to the similarity on the *img* attributes.

Taking the objects with greatest score will favor the *hotel* records.

In this scenario neither the scores are useful to define the concept of similarity, nor the position of each object in the local result sets, i.e. its rank in each answer to the local queries. Ranks could be used to define a new “score” of similarity. Thus, a way to define a *MOST\_SIMILAR* aggregation function is to use the ordinal ranks of each object in the returned lists.

As suggested in [6], a simple yet effective aggregation function for ordinal ranks is the *median* function. This function takes as the aggregated score of an object its median position in all the returned lists. Thus, given  $n$  multimedia sources that answer to a query by returning  $n$  different ranked lists  $r_1, \dots, r_n$ , the aggregated score of an object  $o$  will be  $median(r_1(o), \dots, r_n(o))$ . The aggregated list will reflect the scores computed using the *median* function. The *median* function is demonstrated [6] to be near-optimal, even for *top-k* or partial lists. Then, it is possible to produce an aggregated list even if the different sources return only *top-k* lists as their local answers to the query. For all these reasons, we take the median as the *MOST\_SIMILAR* aggregation function for multimedia objects at the global level.

## 6 Conclusion and Future Work

We presented a methodology implemented in a tool that allows a user to create and query an integrated view of 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. In particular, concerning the query processing, we discussed some cases where multimedia constraints can be expressed in a global query without requiring multimedia processing capabilities at the global level. This is an interesting result, since it upsets the usual paradigm on which mediator systems are based, stating that the *query processing power* of a mediator is greater than the one of the integrated data sources [4]. We will evaluate the effect of this aspect in our future work.

Future work will also be devoted to experiment the tool in real scenarios. In particular, our tool will be exploited for integrating business catalogs related to the area of “tiles”. We think that such data may provide useful test cases because of the need of connecting data about the features of the tiles with their images.

## References

1. G. Amato, C. Gennaro, P. Savino, and F. Rabitti. Milos: a Multimedia Content Management System for Digital Library Applications. In *Proceedings of the 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004)*, volume 3232 of *Lecture Notes in Computer Science*, 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 *IWEB '06: Proceedings of the WWW Workshop in Information Integration on the Web (IWeb)*, 2006.
4. K. C.-C. Chang and H. Garcia-Molina. Mind your vocabulary: Query mapping across heterogeneous information sources. In *SIGMOD Conference*, pages 335–346, 1999.
5. R. Fagin. Fuzzy queries in multimedia database systems. In *PODS '98: Proceedings of the seventeenth symposium on Principles of database systems*, pages 1–10, New York, NY, USA, 1998. ACM.
6. R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *PODS '04: Proceedings of the twenty-third symposium on Principles of database systems*, pages 47–58, New York, NY, USA, 2004. ACM.
7. 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.
8. F. Naumann, J. C. Freytag, and U. Leser. Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615, 2004.