An Architecture for the Integration of Multimedia Heterogeneous Data Sources

A. Chianese, V. Moscato, A. Picariello and A.M. Rinaldi
University of Naples "Federico II" (Italy)
Email: {angchian, vmoscato, picus, amrinald}@unina.it

Abstract—The main focus of the paper is the design of a system which integrates data from heterogeneous multimedia repositories in a single, unified database view, in order to build a multimedia information system capable of integrating data that resides in different data base systems as well as in a variety of non-database data servers. We describe the object oriented data model used as a common model from the wrapper/mediator system, so providing a simple mechanism for enabling the integration while maintaining the independence of the repositories and providing a semantic description of the extracted data.

I. INTRODUCTION

It’s widely agreed that Multimedia is interpreted broadly to mean not only images, video, and audio, but also text and application specific data types (drawings, medical objects, etc.) [5]. It’s natural to model this kind of data by means of objects, in order to capture both the different variety of real data and the different related functionalities. It’s the author opinion that this kind of approach may be useful not only for modelling as single multimedia structured or unstructured repositories, but especially for integrating different data sources in order to give a simple and powerful mechanism for managing the multimedia data. In the literature, several multimedia integration systems have been proposed. In MediaLand [16] the single description of the multimedia data are represented using objects and the correlation among object is obtained by Links. In this way, the authors give a unique conceptual structure for describing the multimedia data, successively clustered in domains called MediaClass. A semantic oriented technology is proposed in Video Anywhere [12], which also allows the interoperability among several information system, especially the video management ones. The core of the system is a Web Content Manager which maintains interactions between video sources and a database containing the metadata description of the different sources, also using by means of an ontology. Another semantic approach is the one described in the InfoSleuth project [3]. In this paper, the heterogeneous data integration is obtained extracting a common view of the semantic content. This approach gives an independence of requests from the information structures, resolving the heterogeneity of data by means of an ontology. InfoSleuth uses KQML, a specific language for communication among agents. Anyway, a more complete and complex system is the one described in the Garlic project [11]. Similarly to the previous project, Garlic models the data with an object-oriented approach, representing in an uniform way the data from different servers. But, differently from the other papers, Garlic provides an efficient query processing and data access layers, for an efficient management of the user queries. The data model is based on Odmg – 93 and define an ODL language for the data definition that is at the base of the Garlic Data Language. A different approach is the one proposed in Impact [13] and is based on the Software Agent Paradigm. Impact Architecture is based on two entities: Agents - software modules created from an user or other agents, having high level functions required from other events - and Impact Servers - representing the services infrastructure created by agents. The Agent Paradigm allows to integrate heterogeneous information using different agents with particular functions and services.

The first contribution of this paper is the integration of the semantic approach with the agent (wrappers/mediators) and database technologies. We first provide an object-oriented model that is used by applications, for interpreting sophisticated user queries and creating semantic execution plans in order to send pieces of queries to the appropriate repositories, and to assembly query results for delivery back to the applications. The paper is organized as follows. In Section II, we outline the data model for multimedia objects management. In section III the system architecture based on Wrapper/Mediator schema for multimedia repository integration is discussed. In sections IV and V, we briefly describe the communication protocol between Wrapper and Mediator. In section VI an example of the system functionalities, with the experimental protocol and related results, is provided. Eventually some concluding remarks are given in Section VII.

II. AN OBJECT-ORIENTED MULTIMEDIA DATA MODEL

We propose an object oriented model (o.o.) for the management of multimedia information. It is well known in the literature that the o.o. technology is effective both for modelling multimedia data [15], [10] and for an efficient query processing [16], [9].

The proposed data model is based on two fundamental concepts:

- MultiMedia Object: it is an entity of the multimedia world. A sunset image or a particular movie are example of multimedia objects.
- MultiMedia Class: it is a set of multimedia objects sharing the same proprieties. The Image or the Video class are examples of multimedia classes.
A generic multimedia object (MMO) can be described by means of a tuple made up of the following elements:

\[ MMO = (OID, OL, AS, AML) \] (1)

- \( O(bject)ID(entifier) \) being the object identifier that distinguishes itself from the other objects,
- \( O(bject)L(ocator) \) being the physical pointer to locate the actual source of the multimedia data so providing a fast and direct access to data,
- \( A(tributes)S(et) \) being the set of attributes (DB Attributes, IR features or CBR features) instances that describes the object,
- \( A(available)M(ethods)L(ist) \) being the list of available methods defined on the object. Each multimedia object requires the definition of these elements throughout its lifetime within our system.

In a similar manner a generic multimedia class (MMC) can be described by means of a tuple made up of the following elements:

\[ MMC = (|OID|, |OL|, AS, AML, CR) \] (2)

- \( |O(bject)ID(entifier)| \) being the domain of object identifiers that distinguish unambiguous the multimedia objects,
- \( |O(bject)L(ocator)| \) being the domain of the physical pointers to locate the actual sources of the multimedia data,
- \( A(tributes)S(et) \) being the set of attributes (DB Attributes, IR features or CBR features) defined on the class,
- \( A(available)M(ethods)L(ist) \) being the implementation of available methods defined on a particular object and
- \( C(lass)(R)elationship \) being the set of relationships between the class being specified and the other existing classes [14].

We also provide the usual class relationships: \( \text{SPECIALIZATION(GENERALIZATION)}(OF)(\cdot) \), the new class A is defined as a specialization of an other class B, in this way A inherits all attributes and methods defined on B (the new class A is defined as a generalization of one or more existing classes, for example B and C, all common attributes and methods of B and C are defined on A); \( \text{SET_OF(\cdot)} \), the new class A is defined as a set of one or more existing classes, for example B and C. In this manner B and C became automatically attributes of the class A; \( \text{OF(\cdot)} \).

Example 2.1 (Object Relationships): The Video class can be seen as a \( \text{SET_OF(\cdot)} \) the Video Scene class, or, the Web Page class can be seen as a set of the Image and Text classes. The Image class can be also seen as \( \text{SPECIALIZATION_OF(\cdot)} \) Photo and Painting classes.

As in the standard o.o. terminology, a base class is also called \( \text{primitive} \), while we define \( \text{complex} \) a class obtained by means of the above defined class-relationship constructs.

Example 2.2 (Multimedia Class Hierarchy): In the figure 1 an example of the application of the class-relationship “rules” for multimedia class definition is provided. The class-relationship concept allows to create a multimedia classes hierarchy [14]. In a such a scenario, a multimedia object is an instance of a particular multimedia class; it is obtained specifying the single values of class attributes.

FIG. 1. CLASS DIAGRAM

III. SYSTEM ARCHITECTURE

The proposed architecture is based on the classic Mediator-Wrapper schema, also used in [4], and tries to satisfy the main requirements of a multimedia database management system [6] [17] [1]. In this kind of approach, the wrapper examines the several multimedia repositories and send the mediator an appropriate description of the related information in terms of multimedia classes. From the other side, the mediator receives and organizes these information in order to create a single view on all repositories in order to satisfy the user queries processing.

A. Mediator

The architecture of the Mediator is shown in figure 2. The user queries are submitted by means of a Multimedia User Interface and the related query results are shown, according to the communication protocol (see sections IV and V). The Wrapper Interface module picks up the “meta-information”, in terms of multimedia classes related to the different multimedia repositories, and the related query results. The set of fundamental multimedia classes managed by the system are stored in the Standard Class DB. The user queries are processed by means of the Query Analyzer, whose results and related information are are stored in the Query DB. On the contrary, the list of effective class-attributes and of all the available methods, for each multimedia repository, is stored in the Metadata DB. In this area the syntactic heterogeneity is usefully solved for query processing. The queries are then “taken and compiled” by means of the Query Engine which sends the appropriate wrappers the given query. The partial and global query results are then stored in the Result DB. The results are managed by the Result Handler and the Top-K Selection module, which analyzes the results and chooses, by means of appropriate strategies, analyzing the Metadata DB content, the best K results and reports them to the Object Recovery module.
Eventually, a CLASSIFIER module takes the repositories class-description from the wrappers and, by means of a STANDARD MULTIMEDIA THESAURUS, tries to link each multimedia class to one of the standard classes contained in the STANDARD CLASS DB. If the classification process is successful, the class attributes and the methods are then stored in the METADATA DB, while activating the EXCEPTION HANDLER in the opposite. A SEMANTIC MANAGER is also used after the classification stage, in order to associate the repository data with a semantic meaning to be stored in METADATA DB. To these aims, the module uses the MULTIMEDIA KNOWLEDGE BASE, by which it is possible to associate the repository information with the related semantic meaning, trained by means of some multimedia objects’ features.

The system is also provided with an advanced CONFIGURATION MANAGER module. It is used to define, during the initialization phase of the system, the standard classes structure (e.g. selection of standard multimedia classes and of the related attributes and methods) and the system policies (e.g. events managing, Top-K selection strategy, user authentication, etc...). In a such way, the system architecture is fully flexible respect to different application environment.

For some multimedia classes, the use of certain standard methods is necessary. For example, the possibility of executing on the attributes of a relational database classical “sql queries” is fundamental. To this aim, in our implementation there exist special methods named “default methods”. The CLASSIFIER expects that such method is defined in multimedia class. If a multimedia class is not provided with this method, the Wrapp-
which must be exported for the class definition. This interface is also called every time there are some errors in the wrapper automatic tasks.

We want to notice that, in a similar manner to the mediator standard class definition, the Wrapper Administrator has the possibility of creating complex multimedia classes from one or more already classified classes, by means of application of class-relationship constructs. In this case the new class must be classified by means of a matching with the possible system standard complex multimedia classes.

C. A Use Example

To best understand the described system we provide an use example which describes the information flow involved.

In the most general case a Multimedia Query is a mixed query which combine textual and multimedia information. Referring to a given multimedia class, the following queries can be expressed: (i) queries with “where” clause; (ii) keyword-based queries; (iii) queries with conditions on multimedia contents (content-based query); (iv) queries with Multimedia Objects (query by example).

The first class of queries can be solved by the classical SQL approach, the second class with an Information-Retrieval approach, while for the last ones the use of techniques for multimedia data management is required. A multimedia query is submitted to the system by means of the USER INTERFACE. The query result is the set of multimedia objects, contained in the repositories, that satisfy the user requests. During a query processing the user has to specify the wanted object-type assigning appropriate weights to the different query components (multimedia class, class attributes, class methods); such weights influence the TOP-K of results.

From the USER INTERFACE a query undergoes the mediation by means of the QUERY ANALYZER which identifies the set of multimedia classes, standard methods, query parameters and query weights necessary to the query processing. After this stage the information about user query are stored in the QUERY DB. At this point the scheduled queries in the QUERY DB are sent to the QUERY ENGINE. This module accesses to the METADATA DB tables related to the standard classes specified in the query. For each wrapper the query is “compiled” using to the attributes and methods names contained in the METADATA DB and the possible parameters inserted by users. The related query-request is sent to the appropriate wrappers.

A given wrapper receives, by means of MEDIATOR INTERFACE, the query-request from Mediator and enables, after a verify processing on the present class, the ANALYZER in order to select the appropriate modules for query solving. In the case of query by example the ANALYZER also requests to mediator the file contained the user query-object. After this preliminary stage the QUERY EXECUTOR picks up all query information and executes the query by the WRAPPER DB. The results are then sent to Mediator.

After the query processing stage the TOP-K module collects and interprets the query results in order to chose, by the use of appropriate strategies [5], the best K ones communicating them to the OBJECT RECOVERY. Eventually this module has the work of, from a side, retrieving, from appropriate wrappers, the wanted multimedia objects, and, from the other one, sending these results to USER INTERFACE. We want to notice that such module is provided with a BUFFER STORAGE AREA for object cashing.

IV. WRAPPER-MEDIATOR COMMUNICATION

In the communication flow with Mediator, a wrapper is interested in: (i) communicating its entry into the system and requesting to the Mediator a new registration; (ii) communicating the presence of a Multimedia Class and requesting to the Mediator its insertion; (iii) communicating possible updating or deleting of Multimedia Class; (iv) communicating the query results; (v) requesting possible query parameters.

In the case of a updating or deleting request, if the involved multimedia class is used in user query, the Mediator locks such class delaying the wrapper operation. The communication is performed by means of XML-files. In the following the main communication procedures between wrapper and mediator are shown.

A. Multimedia Class Insert

In the algorithm 1 the procedure for inserting new multimedia class is shown. In the example, we refer to a an “Image” class. We notice that the default attributes “ObjId” and “ObjLocator” have to be always defined, moreover, the “ClassId” attributes is automatically generated by wrapper to reference a given multimedia class. The couple (Wrapper Identifier, Class Identifier) is used by mediator to access to a multimedia class.

B. Semantic Management

In the algorithm 2 the procedure for the management of objects semantic is shown.

The division in standard classes of the multimedia objects using the CLASSIFIER and the STANDARD MULTIMEDIA THESAURUS modules, allows us to create some “classification ontologies” in order to discover hidden semantic associations among the multimedia objects. In other terms such classification ontologies, opportunely organized for “semantic multimedia domains”, describe, by means of the use of a MULTIMEDIA SEMANTIC NETWORK, the semantic association among the object semantic concepts.

For example, it is easy to think that the discovered semantic concepts “dog” and “cat” are part of the same “pets” classification ontology referring to the “animal” semantic domain. Using the discovered semantic objects associations, it is possible to make more accurate the query processing. The TOP-K SELECTION module takes into account the fact that two or more objects can have similar semantic concepts in the choosing of best query results.
Algorithm 1 Multimedia Class Inserting Procedure

- The wrapper sends to the mediator the new class in the following manner:

```
< MultimediaClass Name = Image ClassId = 00001
ClassId = null/>
```

- The mediator tries to classify the new class; it communicates to the wrapper the following message:

```
<InsertAck ClassId = 00001>
<ClfAck Status = NOK>
<InsAck Status = OK>
</InsertAck>
```

- In the case of correct classification and missed inserting (e.g. the Object Identifier is already used by the same wrapper):

```
<InsertAck ClassId = 00001>
<ClfAck Status = OK>
<InsAck Status = NOK>
</InsertAck>
```

- If such communication has a success according to the described protocol, then the wrapper will be able to send the repositories multimedia data. Such data will be extracted:

```
< MultimediaObject ClassId = 00001
Attributes >
<Attribute Name = ObjId Type = String>
<Attribute Name = ObjLocator Type = String>
<Attribute Name = Shape Type = String>
<Attribute Name = Color Type = String>
</Attributes>
<Methods>
<Method Name = FindSimilarity>
<Parameter Type = File>
</Methods>
</ MultimediaObject>
```

Algorithm 2 Semantic Management Procedure

- After the classification stage the wrapper sends to the mediator the objects set from which a semantic concept has to be extracted:

```
< MultimediaObject ClassId = 00001
Elements >
<Element>
<ObjId Value = WRA1001>
<Attribute Name = ObjId Type = String>
<Attribute Name = ObjLocator Type = String>
</Elements>
```

- The mediator, by means of Multimedia KNOWLEDGE BASE, determines, for each object, the semantic concept with the highest association probability. Using ontologies, the associations among similar semantic concepts are detected.

C. Query Results

The query result is an other fundamental message in the communication flow between wrapper and mediator. In the example, we suppose that the query request is executed on a multimedia object provided by “FindSimilarity” and “SqlWhere” methods. The procedure for the exchange of query results is shown in algorithm 3. We notice that the SqlWhere method returns exact (0 or 1) matching scores, while the FindSimilarity method is characterized by a uncertainty degree.

V. MEDIATOR-WRAPPER COMMUNICATION

In the communication flow with a wrapper, a mediator is interested in: (i) communicating to a wrapper the impossibility to classify a class; in this case the mediator sends to the wrapper the list of available standard multimedia classes in order to assist the wrapper administrator in the re-classification processing; (ii) requesting a query execution; (iii) requesting a sending of one or more files, corresponding to the the best query results.

The communication is performed by means of XML-files. In the following the main communication procedures between mediator and wrapper are shown.

A. Query Request

The query request is a fundamental message in the communication flow between mediator and wrapper. In the example, we suppose that the query is executed on a certain complex multimedia class of Web Page type provided by “FindSimilarity” and “FindWord” methods. The procedure for the query processing is shown in algorithm 4.

Algorithm 3 Query Results Exchanging Procedure

- The wrapper reports to mediator the following query results with the related matching scores:

```
<QueryResults ClassId = 00001 QueryId = 1 UserId = 1>
<Element>
<ObjLocator Value = /usr/local/img/1.jpg>
<SqlWhere Match = 0.45 Weight = 0.6>
<SqlWhere Match = 1 Weight = 0.4>
</Element>
```

- The mediator verifies the query presence on the given class and sends the message:

```
<QueryResultAck ClassId = 00001 QueryId = 1 UserId = 1>
</QueryResultAck>
```

- In the case of missed verify:

```
<QueryResultAck ClassId = 00001 QueryId = 1 UserId = 1>
<ClfAck Status = NOK>
</QueryResultAck>
```

B. Objects Request

At the end of a query, the mediator requests to a wrapper of sending the repositories multimedia data. Such data will be shown to the user. The objects exchange is performed by means of the protocol shown in algorithm 5.

Algorithm 4 Query Processing Procedure

- The mediator reports to the appropriate wrapper the query message:

```
<QueryReq QueryId = 2 ObjId = 00002 UserId = 1>
<Methods>
<Method Name = FindWord Weight = 0.4>
<Parameter Type = File>
</Methods>
```

- The wrapper verifies the object presence and sends the message:

```
<QueryReqAck QueryId = 2 ObjId = 00002 UserId = 1>
</QueryReqAck>
```

- In the case of missed verify:

```
<QueryReqAck QueryId = 2 ObjId = 00002 UserId = 1>
<ClfAck Status = NOK>
</QueryReqAck>
```
VI. THE SYSTEM FUNCTIONALITIES

In the following a complete example of system operation is shown.

A. Wrapper-Side

We suppose the presence of two multimedia repositories containing some scenic (sunsets, deserts, coasts, etc...) and animals (birds, fishes, mammals, etc...) digital images. In particular the first repository is a MySql 4.0.16 Relational Database running on a Linux RedHat 7.3 operating system. The second repository is an Access XP Relational Database running on a Windows XP operating system.

The wrapper functionalities are provided with the USER INTERFACE. It is possible to: (i) registry a new wrapper and configure the wrapper for repository communication; (ii) create a multimedia object; (iii) set communication parameters with mediator.

The definition of the new “Image” multimedia class, with its attributes and methods, is performed by the OBJECT CREATOR. Such module propose to wrapper administrator the possible class attributes and methods in terms of database tables, table fields and stored procedure. The class tree structure is after converted in a XML format which is sent to mediator with the related database data, necessary to associate to the object a semantic meaning as described in algorithm 1. Eventually the description of the object is stored in the WRAPPER DB. If classification problems are not present, the wrapper objects are available for user queries. Possible classification errors are solved by the RECLASSIFICATION MANAGER and WRAPPER ADMINISTRATION modules.

Moreover we suppose that each wrapper is provided with an ANIMATE VISION [7], [8] module in order to implement the FindSimilarity method defined on an image-class object. The application of this matching algorithm returns a number in [0,1] interval which indicates the similarity degree between two images. The matching is not applied on the whole images, in [0,1] interval which indicates the similarity degree between two images. The matching is not applied on the whole images, in [0,1] interval which indicates the similarity degree between two images. The matching is not applied on the whole images, in [0,1] interval which indicates the similarity degree between two images.

The query processing to/from mediator is managed by the QUERY EXECUTOR as described in algorithm 3. Such module uses the information in WRAPPER DB in order to translate mediator “SQLWhere” requests in the local DBMS SQL format and the ANIMATE VISION module to retrieve the similarity measures between the target image and the DB images. Eventually the query results and the related scores are sent to mediator.

B. Mediator-Side

We suppose the the mediator is able to manage a unique standard class called “Image”. The mediator functionalities are provided with two different graphical interfaces. By means of the USER INTERFACE, it is possible to: (i) submit a multimedia query; (ii) view the query results; (ii) set some communication parameters in order to optimize the transmission flow towards the user.

By means of the ADMINISTRATOR INTERFACE shown in figure, it is possible to: (i) define the standard classes managed by the system; (ii) define the system policies (e.g., events managing, wrappers managing, Top-K strategies, data flow security, etc...); (iii) set general parameters (multimedia thesaurus, possible object resolutions for devices transmissions, possible users authentications, etc...).

All mediator functionalities are also provided with Web Services technologies and can be integrated in a more general architecture.

The main task of mediator is to classify the classes originated by the two wrappers. If the classification has a success, the mediator inserts the new class in the METADATA DB. The SEMANTIC MANAGER tries to associates a semantic concept to the wrapper objects and the semantic associations with the other objects. The semantic concept is obtained for multimedia semantic domain (image category) and by object feature values (shape, color, texture), choosing the association with the maximum probability. The semantic object relationship are obtained by using some classification ontologies.

The query processing is, as already seen, carried out in three phases: (i) by means of USER INTERFACE all data (standard class, attributes and methods constraints, Top-K dimension, query weights), necessary to the query execution, are picked up and stored in the QUERY DB; (ii) the query, stored in the QUERY DB, is compiled by the QUERY ENGINE and sent to wrappers; (iii) the query results are picked up form Wrapper and, by means of the TOP-K SELECTION [5], the best ones are shown to the user. During the download, as specified in the user settings, the object resolution is adapted to the user device type (palmar, mobile phone, PC, etc...).

C. Experimental Results

In the following some user query cases are reported. The query processing is performed by means of the FindSimilarity and SQLWhere methods. In the tables I and II the main environment features of our experimentation protocol are listed.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>EXPERIMENTATION SET UP I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediator</td>
<td>Wrappers</td>
</tr>
<tr>
<td>2 CPU (2.5GHz 1Gb RAM)</td>
<td>1 CPU (2GHz 512Mb RAM)</td>
</tr>
</tbody>
</table>

For each case, the query set up parameters (see table III) and the query results (see figure 4), in terms of multimedia objects, matching scores and wrapper sources, are shown.

We want to notice that, in the second query case, the target image does not belong to any wrapper. In a such case the

---

Algorithm 5: Objects Exchange Procedure

- The mediator reports to the appropriate wrapper an object request:
  `<FileReq File="/usr/local/img/1.jpg"/>
  <FileReq/>

- The wrapper verifies the wanted file presence and sends the message:
  `<FileReqStatus Status=OK/>
  <FileReqStatus Status=NOK>

- if the verify has a success and starts the transfer, or:
  `<FileReq>
  <FileReq Status=NOK>

- in the case of missed verify.
TABLE II
EXPERIMENTATION SET UP 2

<table>
<thead>
<tr>
<th>M-W Connection</th>
<th>Client</th>
<th>Client Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>100Mbps LAN</td>
<td>Pocket PC</td>
<td>WLAN 10Mbps</td>
</tr>
</tbody>
</table>

TABLE III
QUERY PARAMETERS

<table>
<thead>
<tr>
<th>Query Case</th>
<th>FindSimilarity weight</th>
<th>SQLWhere weight</th>
<th>Top-K size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>0.3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.5</td>
<td>5</td>
</tr>
</tbody>
</table>

The mediator does not have semantic information about the query object and the search is performed inside the “sunset” whole category. For this reason, in order to have more accurate results, the FindSimilarity method has, respect to the other query cases, a higher weight.

VII. Conclusions

A multimedia data integration system is presented. It allows a single unified database view from more multimedia repositories. The proposed data model describes a generic multimedia entity as an object belonging to a particular class with similar features and general methods. The system architecture is based on the Mediator/Wrapper schema in order to have a fine description and organization of multimedia data. The user query is processed by ad hoc module and sent to appropriate wrappers. The query results are associated with a semantic meaning using a multimedia semantic network; this approach allows us to put into evidence the concept classes of objects and the relations among them.

The reported preliminary results show the efficiency and effectiveness of the proposed architecture performances.

ACKNOWLEDGMENT

This work has been carried out partially under the financial support of the Ministero dell’Istruzione, dell’Università e della Ricerca (MIUR) in the framework of the FIRB Project “Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)”.

REFERENCES


