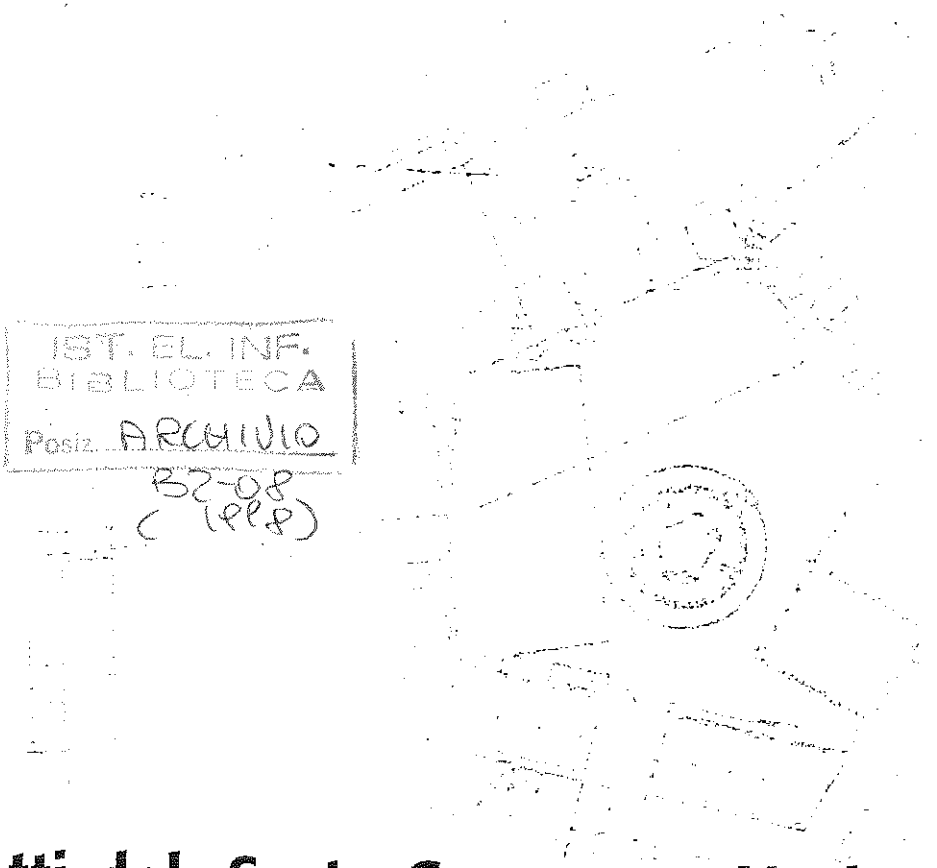

ANCONA 23 - 25 GIUGNO 1998

Atti del Sesto Convegno Nazionale

VOLUME - II -

1998
SEBD

a cura di
Paolo Atzeni
Luca Cabibbo
Maurizio Panti

SISTEMI EVOLUTI PER BASI DI DATI

SEBD '98

Comitato di Programma

Paolo Atzeni	*Università di Roma Tre (Coordinatore)*
Maristella Agosti	*Università di Padova*
Antonio Albano	*Università di Pisa*
Sonia Bergamaschi	*Università di Modena*
Luca Cabibbo	*Università di Roma Tre*
Silvana Castano	*Università di Milano*
Franca Garzotto	*Politecnico di Milano*
Sergio Greco	*Università della Calabria*
Gianni Mainetto	*CNUCE-CNR Pisa*
Michele Missikoff	*IASI-CNR Roma*
Maurizio Panti	*Università di Ancona*
Silvio Salza	*Università di Roma "La Sapienza"*
Fabio Schreiber	*Politecnico di Milano*
Letizia Tanca	*Università di Verona*

Struttura Organizzativa

Presidente

Valeria de Antonellis	*Università di Brescia*

Tutorial

Luigi Palopoli	*Università della Calabria*

Rapporti Industrie Locali

Maurizio Panti	*Università di Ancona*

Comitato Organizzatore

Maurizio Panti	*Università di Ancona (Coordinatore)*
Alessandro Cucchiarelli	*Università di Ancona*
Claudia Diamantini	*Università di Ancona*
Luca Spalazzi	*Università di Ancona*
Salvatore Valenti	*Università di Ancona*

INDICE

VOLUME I

BASI DI DATI E WORLD WIDE WEB

Accesso a Basi di Dati via Web
N. Aloia, C. Concordia 3

The Araneus project: extending database techniques to the World Wide Web
G. Mecca, P. Merialdo, A. Masci, G. Sindoni 19

Using WG-Log to represent semistructured data: the example of OEM
B. Oliboni, L. Tanca, D. Veronese 35

Extending temporal database concepts to the World Wide Web
F. Grandi, M. R. Scalas 53

LOGICA DESCRITTIVA

Utilizzo della logica descrittiva per l'estrazione di proprietà terminologiche e strutturali complesse
A. Bonifati, L. Palopoli, D. Saccà, D. Ursino 71

Extending semi-structured data
D. Calvanese, G. De Giacomo, M. Lenzerini 87

Exploiting schema knowledge for the integration of heterogeneous sources
S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M. Vincini 103

DATA WAREHOUSING E INTEGRAZIONE DI BASI DI DATI

Un quadro metodologico per la costruzione e l'uso di un data warehouse
L. Cabibbo, R. Torlone 123

Progettazione concettuale di Data Warehouse da schemi logici relazionali
M. Golfarelli, S. Rizzi 141

Information integration on multiple heterogeneous data sources
S. Castano, V. De Antonellis, S. De Capitani di Vimercati, M. Melchiori 155

Una misura entropica per la valutazione di cluster concettuali
C. Diamantini, M. Panti 171

VOLUME II

WORKFLOW E TRANSAZIONI

On concurrency management in temporal relational databases
C. De Castro 189

Deterministic and nondeterministic declarative semantics for active rules
S. Flesca, S. Greco 203

Modeling distributed transactions as mobile agents in a distributed heterogeneous database system
A. Di Stefano, L. Lo Bello, C. Santoro 219

WERDE: a pattern-based tool for exception design in workflows
F.Casati, S.Castano, M.G. Fugini, I. Mirbel, B. Pernici 237

DATI MULTIMEDIALI E VISUALIZZAZIONE

Strutture dati e algoritmi per ricerche nearest neighbor incrementali
A. Lumini, D. Maio 255

Retrieval of multimedia data by similarity in the HERMES DBMS
G. Amato, G. Mainetto, P. Savino 271

Interfacce adattative per WWW: il caso di IDL
M. F. Costabile, F. Esposito, G. Semeraro, N. Fanizzi, S. Ferilli 289

Una tecnica visuale per identificare problemi di usabilità in applicazioni ipermediali
M. F. Costabile, M. Matera, G. Piepoli, E. Pullì 307

DATA MINING

Incremental refinement of association rule mining
E. Baralis, G. Psaila 232

BASI DI DATI OBJECT-ORIENTED

Reasonig about set-oriented methods in object databases
G. Guerrini, I. Merlo 343

Specifica diagrammatica di schemi OOBD: graph grammar e parser per TQD++
M. F. Costabile, M. D'Avena, M. Missikoff 359

Tipi e moduli nel linguaggio Fibonacci
G. Ghelli, F. Nanni, G. Puglielli, A. Albano 377

Il gestore di oggetti persistenti del sistema Fibonacci
A. Albano, M. Pieragnoli, D. Dallagiacoma 399

Retrieval of multimedia data by similarity in the HERMES DBMS[1]

Giuseppe Amato*, Gianni Mainetto+, Pasquale Savino*

*IEI-CNR Via S. Maria 46 - 56126 Pisa Italy
Email: G.Amato@iei.pi.cnr.it, Email: P.Savino@iei.pi.cnr.it
+CNUCE-CNR Via S. Maria 36 – 56126 Pisa Italy
Email: G.Mainetto@cnuce.cnr.it

Abstract. This paper presents the main characteristics of the MultiMedia Query Language, a query language designed for supporting content-based similarity retrieval of multimedia documents. The query language processor is a component of the HERMES multimedia database system, a DBMS in which both a structural representation of raw multimedia data and an automatically computed description of the multimedia data content are possible. The MultiMedia Query Language is an extension of a traditional object-oriented query language. It allows users to express restrictions on features, concepts and structural aspects of the multimedia database objects. In addition, the language supports the formulation of queries with imprecise conditions. The outcome of a query evaluation is an set of pairs, each one consisting of an object and a measure of the similarity of the object with the criteria specified in the query, sorted by decreasing degree of similarity.

1. Introduction

Content-based similarity retrieval of multimedia documents is one of the most promising technological challenges of short/medium term research. In terms of the information used to represent the content of multimedia data, we can broadly classify the different approaches to content-based retrieval into three categories [Gudivada95]:

- **keyword based**, where the content of the multimedia data is described through annotations provided by users as for example free text or keywords taken from a controlled vocabulary;
- **feature based**, where a set of features is directly *extracted*, i.e. computed, from the machine readable representation of multimedia data and used for the retrieval. Typical features are values that represent either generic information about multimedia data, such as *colour*, *texture*, *shape*, *speed*, *position*, *motion*, etc., or

[1] A preliminary version of this paper has been presented at the 1st East-European Symp. on Advances in Database Information Systems, St. Petersburg, Sept. 2-5, 1997. This work has been partly funded by EU ESPRIT LTR program, project no. 9141, HERMES.

specific information needed by a particular application, such as *face recognition*, *trademarks* [Wu95], *medical image analysis* [Petrakis96]. Feature extraction is either performed through the supervision and support of the user or automatically, which is in some cases computationally expensive and usually application domain specific. Other examples of existing systems that use features extracted from images are [QBIC95] and [Virage96].

- **concept based**, where application domain knowledge is used to perform an *interpretation* of object's content. This interpretation leads to the recognition of *concepts* which are used to retrieve the object itself. Usually this process is application domain specific and may require the user intervention.

In our opinion, the abstraction mechanisms of a data model for multimedia data should allow one to represent both the features and concepts which are intrinsically present in this kind of data. Furthermore, the data model should allow the description of the structure of the knowledge on such data and to combine it with the other, more traditional, representations of data. In a recent paper [Amato97], we have described the main characteristics of a model of this type and its supporting system. The proposed model is Object-Oriented (OO): this is because the OO model is closer to the complexity of the real world than other models. Since multimedia data are a raw machine readable representation of the reality, we believe that the OO model should better allow one to structure the knowledge about such representation.

In this paper, we will mainly describe how queries can be formulated and executed in the Hermes DBMS, a prototype multimedia DBMS supporting a data model with the above characteristics[2]. The query language of the system, called MultiMedia Query Language (MMQL), has the usual expressive power of an OO query language such as, for example OQL [Cattel96]. It has been extended to allow the formulation of queries that can at the same time consider restrictions on features, concepts and the structural aspects of multimedia objects. Furthermore, the language supports the formulation of queries with imprecise conditions. The result of a query is an set of pairs. Every pair is composed from an object and an estimation of its degree of matching of the query. The result of a query is sorted by decreasing order of this degree of matching. We underline that MMQL is a *textual* language that should be considered as an intermediate language, target of a (set of) *query formulation tool*(s) that allows end-users to directly manage multimedia representations.

The paper is organised as follows. Section 2 provides an overview of the multimedia data model adopted from the Hermes DBMS. The multimedia Query Language, which exploits the features offered by the model, is illustrated in Section 3. Section 4 contains examples of queries while in section 5 the reader can find a description of the techniques that we are investigating in the realisation of the prototype query language processor. Section 6 concludes and outlines some open issues and areas for future research.

[2] The collection of technical reports of the Hermes Project can be found at [Hermes].

2. The data model of the Hermes DBMS

The Hermes model can be considered as composed of three layers: the Multimedia Storage Model layer (MSM), which provides constructs to specify how multimedia information is stored in the database and how it can be accessed; the Multimedia Description Model layer (MDM), which allows the identification of relevant portions of multimedia data; and the Multimedia Interpretation Model layer (MIM), which allows one to represent the semantic content of multimedia objects.

A *multimedia data*, at the lowest level of representation, is an unstructured piece of information stored in the multimedia database - taken either from the real world or from other existing multimedia databases. *Raw objects*, in the MSM, identify these multimedia data. Raw objects do not contain any specification regarding internal content and internal structure. They contain information about their physical encoding and the storage strategy used to store them.

One of the aims of interpreting a set of persistent multimedia data is to make explicit their structure and content in order to support their retrieval.

The MDM is used to individuate the objects to be interpreted. In the Multimedia Description Model, the unstructured content of a raw object can be conveniently structured by representing portions of it, and assembling such basic components into complex objects. Objects of the Multimedia Description Model are those that can be retrieved, manipulated, and delivered.

The MIM offers constructs to represent the content of objects of the MDM at two different levels: a) the physical content is represented by extracting physical *features* from objects, b) the semantic content is obtained by associating objects with pre-defined *concepts*.

The next three subsections will describe the main characteristics of the three layers of the model. Further details can be found in [Amato97].

2.1 The Multimedia Storage Model (MSM)

In the MSM, a *raw object* (RO) can represent any fragment of multimedia information. ROs do not contain any description about their semantic content. They just contain information about their *physical encoding* and information on the *storage strategy* used to store them - which means that it is possible to define the place where raw objects are stored (on local disks, remotely) and the type of access that will be used to retrieve them (sequential, striped, etc.).

The storage strategy of an object is a quadruple *<strname, strtype, file_to_str, str_to_file>*. The name of the storage strategy is given in *strname*; examples are *local_file, local_database_object, striped, off-line, remote-url*, etc. *strtype* is the type of the data structure that contains the information about the storage strategy and that allows one to represent and to access the physical content of a RO that uses this storage strategy. *file_to_str* is a function that stores the content of a file in the database using the associated storage strategy and returns a value of type *strtype*.

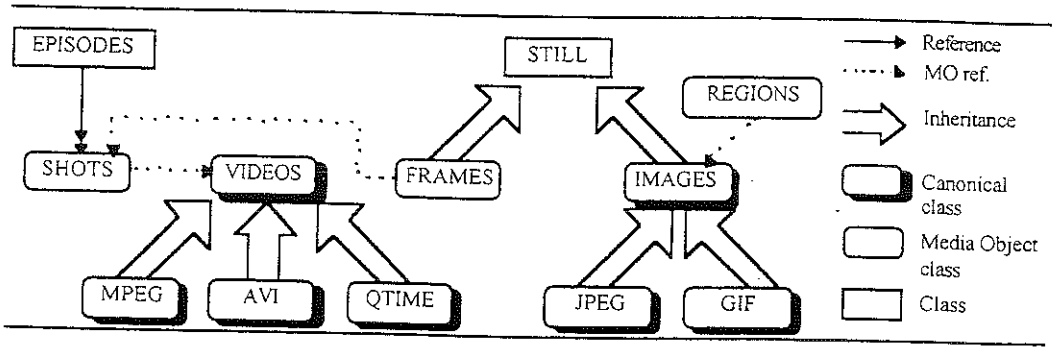

Figure 1. Description model schema

str_to_file is a function that retrieves from the database a multimedia document represented by a value of type *strtype*, and store it in a file.

When a new RO is created, its storage strategy name and the source file should be specified. Then the content can be transparently managed regardless the storage strategy adopted. When the content of a RO is requested, the *str_to_file* function is automatically used. When the content of a RO is updated the *file_to_str* function is automatically used.

For example a RO may represent an image encoded into the GIF format and its storage strategy may indicate that following a particular web URL can fetch the object. Another RO may represent a video encoded into the MPEG format and its storage strategy specifies that it must be stored by using a striping algorithm. Both objects are accessed by a raw object identifier, regardless the details that involve their corresponding storage strategies.

2.2 The Multimedia Description Model (MDM)

The function of the MDM is to provide the mechanisms for defining and manipulating the structure of the information represented by ROs. At the MDM level, no assumption on the semantic content of the documents is made, only their structure is handled. The semantics is provided by the MIM.

The MDM provides three types of objects: the *Canonical Objects* (COs), which represent entire multimedia documents; the *Media objects* (MOs), that represent relevant portions of COs; the *Complex objects* (CXOs), which provide a way of aggregating COs and MOs (as well as CXOs themselves).

For example a CO may represent an image or a video while a MO can represent the region of an image that contains a person. An HTML page containing in-line images and embedded videos can be represented by a CXO that points to a CO that contains the HTML source, to a set of COs that contain the in-line images and to a set of COs that contain the embedded videos. Classes of CO, MO and CXO can be defined to classify different types of documents. Figure 1 shows an example of description model schema.

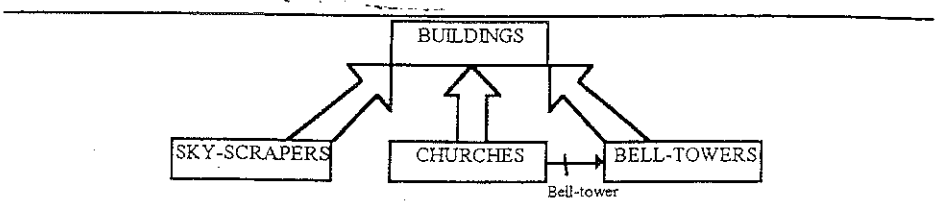

Figure 2. Conceptual level schema

2.3 The Multimedia Interpretation Model (MIM)

The MIM is used to represent the content of COs and Mos. Two different levels of interpretations are considered by the Multimedia Interpretation Model: the *feature level* – using physical properties of the objects – and the *concept level* – according to the semantic content of the objects.

The *feature level* describes the content of objects of the MDM by measuring the values of some of their physical properties, that is the *features*. Examples of features are *color distribution*, *shape*, *texture*, *motion vectors*, etc.

A feature is individuated through a name, an *extractor function* used to calculate the value of the feature in an object, a *similarity function* used to compare feature values, a data structure used to represent feature values, and the name of a class of the MDM that contains the objects which the feature can be extracted from. In the HERMES model is possible to define new features according to user and application needs, providing the previous mentioned components.

The measure of similarity between features is used during document retrieval in order to measure the degree of matching between the query and the retrieved documents. Indeed, the process of retrieving multimedia documents is imprecise: the system does not retrieve the documents that *exactly* match the query – it ranks the documents according to their *similarity* with the query.

The *concept level* describes the semantic content of objects of the description model. It may describe and represent the conceptual entities contained in an object and the relations among entities. This is obtained using an object-oriented model extended to cope with the issues of multimedia document description. *Concepts* are represented both by classes and objects. Each concept associated with a set of *membership functions*. Every function associated with a concept corresponds to a different class of the description model. These functions provide a measure of the degree of recognition of the concept in an object of the description level. Several strategies can be used to define membership function and to calculate the degree of recognition of a concept.
For example, a concept can be recognised in an object because a prototype – representing the concept – is similar to the object; it can be recognised by identifying a particular combination of features; a concept can be recognised because other combination of concept have already been recognised in an object. Figure 2 shows an example of conceptual schema.

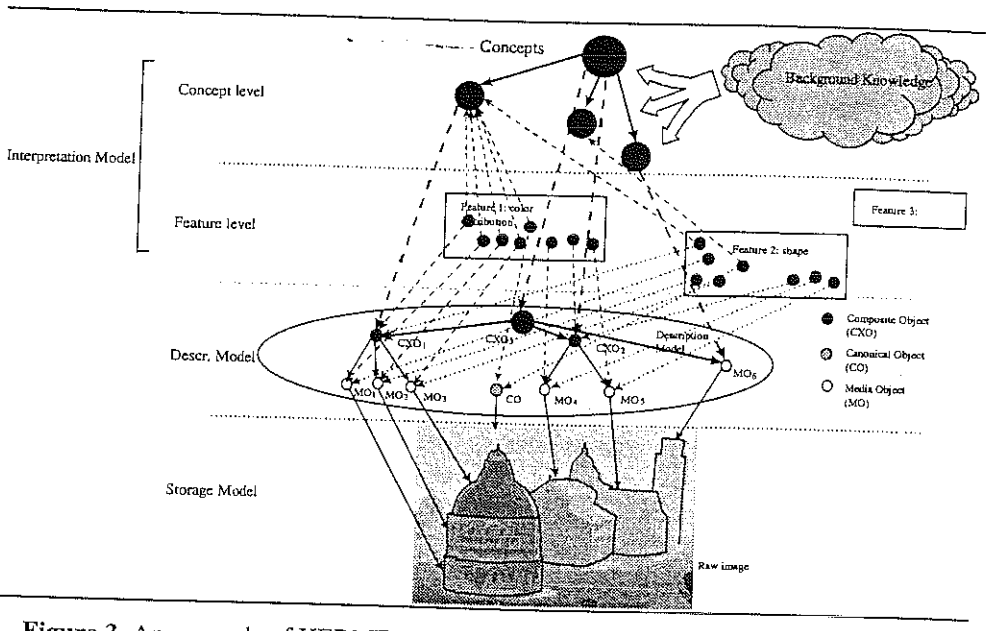

Figure 3. An example of HERMES database

Figure 3 shows all levels of the model used to describe the interpretation of an image. The parts of the image that contain the tower, the Baptistery and the Duomo are considered as the most relevant and are represented at description level as MOs while the entire image is a CO. Feature defined at the feature level are extracted from objects of the description level. These features can be used to recognise concepts at the concept level.

3. The MultiMedia Query Language

The retrieval process in Multimedia DBMS is inherently different from the retrieval process in traditional DBMS. Retrieval in DBMS is based on the exact evaluation of a boolean combination of predicates on attributes. Each attribute has a well defined domain and predicates which can be applied to it. It is possible to exactly determine in a DBMS when the query is satisfied or not. The answer to a query is the set of database records for which the query condition (i.e. the boolean combination of predicates on record attributes) evaluate to "true".

Vice versa, the retrieval of multimedia data consists in determining all documents whose properties are *similar* to those specified in the query {[Cardenas93], [Day95], [Dimitrova94], [Petrakis93], [Petrakis96], [Wu95]}.

In general, a similarity-based retrieval is needed when:

- exact comparison is not possible, it is too restrictive or it may even lead to an empty set results-the data is vague and/or the user is not able to express queries in a precise way;

– ranking of retrieved objects is needed so that the set of retrieved objects can be restricted and/or qualifying objects shown to the user in decreasing order of relevance.

Similarity based queries can be processed if a measure of the *similarity* between each retrieved object and the query (the *matching degree*) can be computed. The result of a query execution is a set of objects ranked according to the value of the matching degree of each object. The imprecision of the match between the query and the retrieved objects is due to the fact that the interpretation of multimedia documents is intrinsically imprecise: the extraction of features and the recognition of concepts may be affected by errors; the result of the interpretation is the value of the feature or the name of the concept associated with the object, plus a *recognition degree* which gives a measure of the quality of the recognition. The most frequently used types of similarity queries are the *range* and the *nearest neighbour* queries. A *range query* has the form: *find all objects that are similar to the desirable query object within a specific distance*; the *nearest neighbour query* has the form: *find the first k closest objects to the given query object*.

Traditional query languages does not provide an adequate support to express similarity based queries. Furthermore, the user usually has only an imprecise knowledge of the characteristics of documents he is seeking and it is also difficult, with features offered by available query languages, to express queries which help in discriminating between relevant and non relevant documents.

Here we propose a query language that has the traditional *select-from-where* syntax and has constructs specialised for multimedia data retrieval. Of course, because of the nature of queries that users would like to express in order to retrieve multimedia documents, the best interface to specify queries would be a graphical interface. The described query language can be considered as the target language of a *query formulation tool* which provides such a graphical interface.

The proposed language allows users to express the following types of restrictions:

- **Features**: The user may express restrictions on the values of *object's features*. An example of a query on features, provided that the features colour, spatial position and motion have been defined, is "Retrieve all videos that have a *red* spot on this *position* that is *moving toward left*".
- **Concepts**: Queries that check the presence of a concept in a multimedia document can be expressed using objects of the concept level. As mentioned above, objects of the concept level are associated with a set of membership functions used to check the presence of a concept in objects of the description level. An example of a query that uses concepts is as follows: "Retrieve all images that contain a *person*"; we suppose that the concept person has been defined and that this concept is associated with a membership function for the class *images* of the description model.
- **Object Structure**: Single media objects as well as multimedia objects are structured, as illustrated in previous section. The query language will allow the user to express restrictions on the structure of the multimedia objects to be retrieved. An example of query that uses the structure is: "Retrieve all *shots* of this *episode* of this *video*".

- **Spatio-temporal aspects**: An important feature of multimedia data is related to the spatial and temporal relationships among different objects. The user should have the possibility to formulate restrictions on the spatial and temporal relationships of the objects to be retrieved. This is possible if specific features have been defined to express object spatial and temporal position. These features also entail defining operations to measure the relative position of two (or more) objects.

Furthermore the proposed query handles the following aspects:

- **Uncertainty**: The query language will allow one to express the uncertainty the user has on some of the restriction formulated and on the preference the user may have on some conditions with respect to others. For example the user may not be certain of the colour of an object, while he is sure of the presence of an object. The values of preference and importance will be used to measure the degree of matching between the query and the retrieved objects.
- **Imprecision**: Similarity and classification operations are intrinsically imprecise. Similarity based retrieval is needed since often exact comparison is not possible, it is too restrictive or it may lead to empty results. Ranking of the retrieved elements is needed in these cases so that the set of the retrieved elements can be restricted and/or qualifying elements shown to the user in decreasing order of relevance. To cope with these issues, the logical operators of the query language should deal with *recognition degrees* instead of *booleans*. Query executions should always return an ordered set of objects. In the ordered set each element is associated with a value that represents a measure of the degree of relevance of the element and the ordering is made with respect to that value.

3.1 The SELECT-FROM-WHERE construct

The MultiMedia Query Language (MMQL) offers both the standard functionality of an OO query language and the constructs to express the previous computational needs.

A query has the typical select-from-where structure:

```
<Query>::= select <select-clause>
           from   <from-clause>
           where  <condition>
```

where:

- `<from-clause>` specifies the set of objects that are candidates to be returned by the query. It may be one of the following elements: a *concept level class*, a *description level class*, a *subquery*, the *union* `U`, the *intersection* `I`, or the *cartesian product* `X` of two queries:

```
<from-clause>::=    <Class>    |  <Dclass>          |
                    <Query>       |
                    U (fs_1, w_1, fs_2, w_2)|
                    I (fs_1, w_1, fs_2, w_2) |
                    X (fs_1, w_1, fs_2, w_2)
        where fs_1,fs_2∈<from-clause>
```

Union, intersection and-cartesian product operations *do not follow the boolean logic*. For example, σ (Q_1, w_1, Q_2, w_2) represents the union of queries Q_1 and Q_2 with a relative importance w_1 and w_2 respectively. This operation will perform the union of the result of Q_1 and Q_2; the degree of matching of each object will depend its degree of matching in Q_1 (resp. Q_2) and the relative importance w_1 (resp. w_2).

- <select-clause> has the form:

  <select-clause> ::= [*n-hits*]<select-list>

  where *n-hits* indicates that only the first n-hits should be shown as the result and <select-list> is an expression that specifies what a query should return for each element of the from clause that has been validated in the query condition. It represents the projection of the query. The query projection is a valid expression based on constants and on paths rooted at <from-clause>.

  For example, if the from clause of the query contains the *class Persons* that as an attribute *Name*, then a valid expression for the select-list could be *Person.Name*.

- <condition> is the query condition. It can be a simple condition or a complex condition. A simple condition can be a precise or imprecise comparison. A complex condition is an expression that may involve simple conditions and other complex conditions:

  <condition>::= <simple-cond> | <complex-cond>

  <simple-cond> ::= <precise-compar> |<imprecise-compar>

  The precise comparison expressions are the usual comparison expressions such as equal, less-than, greater-than.

In the following we will describe the imprecise comparisons and the complex expressions. We will indicate with *v* expressions that evaluates to a generic value, with *o* expressions that evaluates to a generic object, with *do* expressions that evaluates to an object of the description level, with *mo* expressions that evaluates to a media object, with *c* expressions that stands for a concept, with *e* condition expressions, with *w* weights to be associated to conditions when complex condition are specified, with *fv* expressions that evaluate to feature values.

3.2 Imprecise comparison.

Imprecise comparition, as previously stated, returns recognition degree instead of boolean values. These recognition degrees are real values in the range [0 ... 1].

The proposed language basically provides three operators that perform imprecise comparitions. The first checks the similarity between two feature values. The second checks recognition degree of a concept in a multimedia document. The last allows one to infer to what degree a value belongs to a ranked set.

The syntax for these operators is the following:

```
fv_1 sim fv_2        similarity between feature values
do match class       a descr. obj. matches a conceptual class?
do match ref(co)     a descr. obj. matches a conceptual object?
v in <Query>         tests if a value belongs to a ranked set
```

The **sim** operator is used to evaluate the similarity between two feature values. Since the set of features that can be used is not fixed but users can define new features, as we said previously, the behavior of this operator cannot be determined statically. It depends on how the feature, which the specified feature values belong to, has been defined. Actually the evaluation of the **sim** operator corresponds to evaluate the similarity function associated with the feature which the feature values belong to. The **sim** operator will behave in different way if we check the similarity between two colours, between two shapes, between two motion vectors or between two strings.

The **match** operator measures the recognition degree of a concept in a description object *do*. There are two forms of the **match** operator. One should be used for concepts represented by classes, the other for concepts represented by objects. The behaviour of this operator cannot defined statically since evaluating the **match** operator corresponds to evaluating the membership function of the concept corresponding to the description model class which the description model object *do* belongs to. As we said previously, the membership functions should be provided by users when they define new concepts. Two different concepts may be associated with different membership functions.

The **in** operator tests to what degree a value belongs to a ranked set. It actually returns the rank associated with the specified value in the ranked set. It corresponds to the traditional SQL **in** operator with the difference that it works whit ranked sets instead of traditional sets and returns recognition degrees (i.e. ranks) instead of booleans.

3.3 Complex conditions

Simple conditions can be combined into complex conditions through the use of **and**, **or** and **not** operators. In the following we consider that two expressions, `e_1` and `e_2` have to be combined. Expression `e_1` (resp. `e_2`) has a relevance `w_1` (`w_2`). The relevance allows one to specify the weight to be assigned to each expression. It takes into account the uncertainty that users may have regarding some of the conditions they are expressing (for example they want to express that it is much more important that the retrieved images contain a church rather than of a bell tower).

The proposed operators are the following:

```
e_1 [,w_1] and e_2 [,w_2]
e_1 [,w_1] or  e_2 [,w_2]
not e
```

The query language does not impose any constraints on the method to be used to compute the recognition degree of the complex expressions. This is a task of the query

processor, which is not described in the paper. However, various approaches can be followed, such as the adoption of fuzzy logic, probabilistic logic, or the probabilistic model of Information Retrieval [Salton89].

The language is not tied to a specific approach whose validity has still not been demonstrated. The query language uses constructs that deal with imprecision and it manages recognition degrees, but we do not restrict it to a specific approach. However, a specific approach must be defined when the language is implemented, since its choice affects implementation and optimisation techniques.

The intuitive meaning of the **and** and **or** operators should be preserved, irrespective of the implementation: the use of the **and** operator means that both recognition degrees of the terms should be high; the use of the **or** operator means that at least one of the recognition degrees should be high. The difference between the actual behaviour and the intuitive behaviour of the language can be considered as a matter of precision and it is often subjective. Since the information is also intrinsically imprecise, the user may also use the language without knowing the actual implementation of constructs. Users may only use their intuitive ideas of the behaviour knowing that the results are affected by a certain degree of imprecision.

3.4 Selectors

The proposed language supports all classical expression of an object oriented query language and constructs specific for the adopted multimedia model. In particular, three specific selectors are offered to cope with feature extraction, recognition degrees and structure of multimedia documents in addition to classical selectors for accessing fields of structured values and for evaluating objects' methods. The first accesses the value of a particular feature in a multimedia document. The second accesses the recognition degree of a candidate value of the query, the third accesses the object which a media object has been extracted from:

ref(*o*)	gets the object identity of an object
o.**feature** (*fid*)	feature value of a description object
v.**rec-degree**	recognition degree of a value
mo.**part-of**	original object of a media object
v.attribute	field of a structured value
o.method(*args*)	method evaluation of an object

Given an object of the description level, it is possible to access its feature value corresponding to the feature *fid* using the **feature** selector. This would correspond to evaluating the feature extraction function associated with the specified feature. However notice that generally it is not necessary to evaluate that function since feature extraction is done when objects are inserted into the database, in order to index them, and the feature values can be successively obtained simply accessing some indexing structure.

The **rec-degree** selector is used to access the recognition degree of values. During the query processing, the values specified in the <from-clause> are validated

using the <condition> specified for the query. This validation process corresponds to check the <condition> for each value of the <from-clause>. At the end each value is associated with its recognition degree. The **rec-degree** selector accesses this recognition degree. For instance the following query:

```
select I
from IMAGES as I
where I.feature(color) sim red
  and I.rec-degree > 0.7;
```

evaluates to all images whose colour is similar to red with a degree greater than 0.7.

Since the condition can be a complex condition, the **rec-degree** selector returns the recognition degree of a value limited to the context in which the selector is used. For instance the following query:

```
select I
from IMAGES as I
where ( I.feature(color) sim red
        and I.rec-degree > 0.7 )
  and ( I.feature(shape) sim circle
        and I.rec-degree > 0.5 );
```

evaluates to all images whose colour is similar to red with a degree greater than 0.7 and whose shape is similar to a circle with a degree greater than 0.5. On the other hand, the following query:

```
select I
from IMAGES as I
where ( I.feature(color) sim red
        and I.feature(shape) sim circle )
  and I.rec-degree > 0.6;
```

evaluates to all images whose colour is similar to red and shape is similar to circle with a joint recognition degree greater than 0.6.

The **part-of** selector is used to access the object of the description model which the specified media object refers to. This can be useful for instance, given a media object that represent a shot or a frame, to access the video which it belongs to.

4. Examples of queries

In the examples of queries that follow the description model schema and concept level schema of Figure 1 and Figure 2 are used.

At the description level we consider videos and images. Videos can be encoded in MPEG, AVI and Quick time. Images can be encoded in JPEG and GIF. We represent the structure of videos in terms of frames shots and episodes. We represent regions of images that contain relevant entities. This is expressed in the description model schema by defining two class of canonical objects: the VIDEOS and the IMAGES

classes. The class VIDEOS has three subclasses of canonical objects: MPEG, AVI and QTIME classes, each one corresponding to a different video encoding. The class IMAGES has two subclasses of canonical objects: the JPEG and GIF classes. We represent shots and frames as media objects. These media objects are grouped in the SHOTS and FRAMES classes of media objects. Shots are extracted form videos. Frames are extracted from shots. Set of shots that form an episodes are represented by objects of the EPISODES class. Relevant regions of images are represented by media objects of the REGIONS class. Frames and images are both considered as still images so the FRAMES and IMAGES classes are subclasses of the STILL class.

At the concept level we suppose to model important buildings. Actually we want to manage information about sky scrapers, churches and bell towers. In the concept level schema the BUILDINGS class has three subclasses: the SKY-SCRAPERS, CHURCHES and BELL-TOWERS classes. The objects of the CHURCHES class that represents churches that have an important bell-tower are associated with the object of the BELL-TOWER class that represents such a bell-tower. We suppose that classes and objects are all associated with membership functions that tests if objects of the description level contain the concept they represent.

We suppose that the feature *shape*, the feature *color* and the feature *position* are defined. They are extracted from objects of the class REGIONS of the description level. The functions *left-to* and *right-to* have been defined to compare positions.

4.1 Query 1

Features can be used to retrieve multimedia documents that contain concepts that have not been defined. For instance in our example the concept *sun* has not been defined. However we can search for regions that contain the sun searching for regions that are circular and red. Therefore the query that we should express is "*retrieve the ten best regions that are circular and mostly red*". Let's suppose that we are very confident on the fact that the region we are looking for is red; on the other hand we are less convinced that the region should be circular. In the query we use the two constants *circle* and *red*, that we suppose have already been defined. They represent the feature values corresponding to a circle and to the red colour. The following query accomplishes this task:

```
select 10 ref(R)
from REGIONS as R
where R.feature(color) sim red, 0.7
  and R.feature(shape) sim circle, 0.3;
```

The user's uncertainty has been expressed by associating different weights to the two clauses of the condition. We used 0.7 for the similarity to the *red* color and 0.3 for the similarity to the *circle* shape. Notice that, if we wanted to define the sun concept, the previous query could be used to build a membership function for such a concept.

4.2 Query 2.

Let us suppose that we want to "*retrieve all images that contain churches with their bell tower*". We want to privilege the fact that in the image there is the church instead of the fact that there is the bell tower.

The query performs a cartesian product between the IMAGES and the CHURCHES classes then it keeps just the rows in which the image matches both the church and its bell tower:

```
select ref(I)
   from CHURCHES as C, IMAGES as I
   where I match ref(C), 0.6
     and I match C.bell_tower, 0.4;
```

The previous query mixes information extracted from multimedia data and conceptual information: the association between a church and its bell tower is a concept level information; the fact that both appear in the same images is inferred from multimedia data.

4.3 Query 3.

Let's suppose that the concept *churches* is associated with a membership function that recognises it in video frames. We want to "*search for all shots that contain a church*". In order to retrieve shots that contain a church we can use the *churches* concept to retrieve frames that contain a church, then we can use the information on the structure of videos to retrieve the corresponding shots. In addition let's suppose that we want to be quite accurate so we want that the recognition degree should be greater than 0.7. The following query returns what we need:

```
select ref(S)
from SHOTS
where ref(S) in (select F.part-of
                 from FRAMES as F
                 where F match CHURCHES)
  and S.rec-degree > 0.7;
```

4.4 Query 4.

Spatial queries can be expressed by using the feature *position*. Let us suppose that we want to "*retrieve the 10 best fitting images that contain a circular region on the left of a rectangular region*". Let us suppose that we have already defined the two constants *circle* and *rectangle* that respectively represent the feature values corresponding to a circle and to a rectangle. This query can be expressed as follows:

```
select 10 ref(I)
from IMAGES as I, REGIONS as C, REGIONS as R
where
```

```
C.part-of = ref(I) and
R.part-of = ref(I) and
C.feature(shape) sim circle and
R.feature(shape) sim rectangle and
left-to(C.feature(position),
    R.feature(position));
```

In the query the uncertainty is introduced when the shapes of the regions are compared with that of circles and rectangles. The other clauses of the condition are in fact boolean conditions.

4.5 Query 5.

The following query uses at the same time concept level, feature level and description level information. Let us suppose that we want to retrieve *"all buildings whose shape is similar to that of the Empire State Building"*. Let us suppose that objects of the concept level class BUILDINGS have a field *name* that contains the name of the building. We want to privilege the fact that a region is a building instead of the fact that it is similar to the Empire state building.

The query should first retrieve all regions that match the Empires State Building. Then all regions that match some building and whose shape is similar to that of the regions containing the Empire State Building are retrieved. Since we want to give more importance to the fact that a region is a building then it is similar to the Empire state building, we use the weight 0.7 for the first clause and 0.3 for the second:

```
select ref(B)
from REGIONS as R1, BUILDINGS as B
where
  R1.feature(shape) sim any(
    select R.feature(shape)
    from REGIONS as R
    where
      R match (
        select unique ref(SS)
        from SKY_SCRAPESR as SS
        where SS.name='Emp. St. Build.'), 0.3
  and
  R1 match ref(B), 0.7;
```

5. Query processing issues and implementation techniques

We are currently working at the design and implementation of the query language processor for the described query language. In the following we will briefly summarise the techniques we are investigating.

MMQL allows users to express range queries and nearest neighbour queries using weighted similarity predicates. An efficient processing of this kind of queries is still an open research issue. Traditional query processing and query optimisation techniques are not sufficient to cope with them.

For instance consider the following nearest neighbour query: *find the best k objects having (Shape sim 'round') and (Colour sim 'red')*. Each predicate somehow orders objects of the database correspondingly. How can this query be executed without scanning the entire database? Notice that the best k objects for the former predicate might be the worst k for the latter.

[Fagin96] provides a proposal for solving this problem. Supposing that

- the *and* operator is defined as the standard *fuzzy and* operator using the *min* between the scores of the two predicates,
- the predicates are independent,

the suggested approach is to retrieve k' objects from each predicate where $k' = k^{1/m} N^{1-1/m}$. It is shown that retrieving k' objects insures that in the intersection of the results of the two predicates there are at least k objects. In the previous formula k is the number of nearest neighbours for the conjunction, N is the number of objects in the database, m is the number of conditions in the conjunction.

[Chaudhuri96] proposes a strategy to solve range queries in which multiple predicates occur. It uses a notion of *Selectivity Statistics* and an ad-hoc cost model to decide in which order the predicates should be processed. The selectivity statistics, given a range r and an object O_q used as the term of comparison, provide the expected amount of objects whose distance from O_q is smaller than r. The idea is that, if there is a conjunction of predicates, then the best strategy is to begin the computation with the predicate that returns the smallest amount of objects and checking the score of the obtained objects in the other predicates. However the problem of obtaining *Selectivity Statistics* is in most cases a complex task to deal with.

The problem of solving similarity condition with weights has been addressed in [Fagin97]. Provided that the grades g_i and the weights w_i for i= 1,2,...m are in the interval [0,1] and that the weights are normalised, the following formula has been proposed to compute scores for the conjunction of m atomic conditions with weights:

$$f_{(w_1,\ldots,w_m)}(g_1,\ldots,g_m) = \sum_{i}^{m-1} \left[i(w_i - w_{i+1}) \cdot \min\{g_1,\ldots,g_i\} \right] + m \cdot w_m \cdot \min\{g_1,\ldots,g_m\}$$

In order to compute scores for a disjunction of m atomic conditions, a similar formula can be used where the scoring function *min* is substituted by the function *max*:

$$f_{(w_1,\ldots,w_m)}(g_1,\ldots,g_m) = \sum_{i}^{m-1} \left[i(w_i - w_{i+1}) \cdot \max\{g_1,\ldots,g_i\} \right] + m \cdot w_m \cdot \max\{g_1,\ldots,g_m\}$$

6. Conclusions and future work

In this paper we presented a multimedia Query Language that supports (i) partial match retrieval, i.e. there are retrieved all objects which are similar to the query at least with a certain degree; (ii) restrictions on the values of features, the presence of concepts and the structure of objects; (iii) the possibility to take into account user uncertainty on some parts of the query (iv) the possibility to take into account the imprecision of the interpretation of the content of the multimedia object.

The result of a query is a ranked set, that is a set of pairs (object, recognition degree). The recognition degree is a measure of the degree of match between the query and the object.

We consider our future work will evolve along the following main directions:

1. Study how to combine similarity degrees deriving from different features and concepts.
2. The implications of these models with the storage and access of multimedia objects will be studied. Indeed, real application environments will require the storage of many trillions of bytes of data, requiring the use of a storage hierarchy consisting of different layers. This implies that data placement is crucial for effective manipulation of the data and for an efficient retrieval.
3. Completion of the implementation of the system that supports the proposed query language.

7. Bibliography

[Amato97] G. Amato, G. Mainetto, P. Savino, "An Approach to a Content-Based Retrieval of Multimedia Data", *Multimedia, Tools and Application,* Kluwer Academics Publishers, (forthcoming) (a preliminar version is available as IEI-CNR Technical Report N. B4-36-12-96).

[Cardenas93] A. Cardenas, I. Ieong, R. Taira, R. Barker, and C. Breant. "The Knowledge-Based Object-Oriented PICQUERY+ Language", *IEEE Transactions on Knowledge and Data Engineering, 5(4):644--657,* Aug. 1993.

[Cattel96] R. Cattel. "*The Object Database Standard: ODMG-93, Release 1.2*". Norwell, MA, 1996.

[Day95] Y. Day, S. Dagtas, M. Iino, A. Khokhar, and A. Ghafoor. "Object-Oriented Conceptual Modeling of Video Data", In *Proc. of the 11th Int. Conf. on Data Engineering*, Taiwan, pages 401--408, 1995.

[Dimitrova94] N. Dimitrova and F. Golshani. "Rx for Semantic Video Database Retrieval", In *Proceedings of the ACM Multimedia '94*, 1994.

[Fagin96] R. Fagin. Combining Fuzzy Information from Multiple Systems. Proceedings of the ACM-PODS 96, Montreal, Canada, 1996, pp. 216-226.

[Fagin97] R. Fagin and E.L. Wimmers Incorporating User Preferences in Multimedia Queries. Proceedings of the International Conference on Database Theory. Delphi, Greece. Lecture Notes in Computer Science, Vol. 1186, Springer, 1997, ISBN 3-540-62222-5, pp. 247-261

[Gudivada95] V. N. Gudivada and V. V. Raghavan. "Content-based Image Retrieval Systems: Guest Editors' Introduction", *IEEE Computer, 28(9):18--22*, September 1995.

[Chaudhuri96] S. Chaudhuri and L. Gravano. Optimizing Queries over Multimedia Repositories. Proceedings of the ACM SIGMOD 96, Montreal, Canada, 1996, pp. 91-102.

[Hermes] http://www.ced.tuc.gr/Research/HERMES.htm

[Petrakis93] E. Petrakis and S. Orphanoudakis. "Methodology for the Representation, Indexing and Retrieval of Images by Content", Image and Vision Computing, 11(8):504--521, Oct. 1993.

[Petrakis96] E. Petrakis and C. Faloutsos. "Similarity Searching in Large Image Databases", *IEEE Transactions on Knowledge and Data Engineering*, 1996.

[QBIC95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. "Query by image and video content: The QBIC system.", *IEEE Computer, 28(9):23--32*, September 1995.

[Salton89] G. Salton. "*Automatic Text Processing - the Transformation, Analysis and Retrieval of Information by Computer*", Addison-Wesley, Readings, MA, 1989.

[Virage96] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C.-F. Shu. "The Virage image search engine: An open framework for image management.", In *Proceedings of the SPIE 96*, 1996.

[Wu95] J. Wu, A. Narasimhalu, B. Mehtre, C. Lam, and Y. Gao. "CORE: A Content-based Retrieval Engine for Multimedia Information Systems", *Multimedia Systems, 3(1):25--41*, Feb. 1995.