**Henrich, Andreas; Robbert, Günter**

**POQLMM : A Query Language for Structured Multimedia Documents**

Date of secondary publication: 14.12.2023

Version of Record (Published Version), Conferenceobject

Persistent identifier: urn:nbn:de:bvb:473-irb-924236

**Primary publication**

Henrich, Andreas; Robbert, Günter (2001): „POQLMM : A Query Language for Structured Multimedia Documents“. In: Mohand-Said Hacid (Ed.), MDDE'01 : Multimedia Data and Document Engineering ; Proceedings of the First International Workshop on Multimedia Data and Document Engineering, Lyon, France, July 4, 2001, Lyon, 10 pages.

# POQL$^{MM}$: A Query Language for Structured Multimedia Documents

Andreas Henrich
Department for Databases
and Information Retrieval
University of Bamberg
D-96045 Bamberg, Germany

Günter Robbert
Department for Databases
and Information Retrieval
University of Bamberg
D-96045 Bamberg, Germany

**Abstract** *An important research issue with multimedia databases is the formulation of queries to the database. Here the search for structured multimedia documents or relevant parts of it gains more and more significance. Unfortunately text retrieval on its own is not sufficient for the realization of capable search services in this respect. Therefore, an efficient combination of automatic text retrieval, retrieval in meta data and content based retrieval on structured multimedia documents is needed.*

*In the present paper we propose POQL$^{MM}$ as a general purpose query language for an object oriented multimedia database, which supports the combination of text retrieval, content-based retrieval for other media types and traditional fact retrieval on structured multimedia documents. To this end, we describe the significant features of POQL$^{MM}$ and point out the applicability of POQL$^{MM}$ by means of some query examples.*

*Keywords:* structured multimedia documents, query language

## 1 Motivation

Up to now text retrieval dominates the search for documents in many application areas. But the relevance of other media types such as image, audio or video increases steadily. Today multimedia databases usually maintain a large collection of structured multimedia documents. As a consequence, it is not sufficient for a query language for multimedia databases to provide traditional fact retrieval facilities augmented by basic text retrieval techniques. Instead a query language for multimedia databases should permit the exploitation of available meta data as well as content based retrieval facilities on structured multimedia documents considering the structure and the content of the diverse media objects.

In addition to the usual requirements for query languages, such as descriptiveness, orthogonality and extensibility, in our opinion a general purpose query language for multimedia data has to fulfill seven important requirements (cf. [13]). (1) The query language (QL) must allow to search for arbitrary granules ranging from whole documents over intermediate chunks to single media objects. (2) With multimedia data the semantics is usually given implicitly in the media objects, therefore the QL should allow to *extract features* from the media objects potentially describing their semantics. (3) Because of the vagueness in the interpretation of the media objects and in the expression of the users information need *partial match* and *similarity queries* should be facilitated. (4) Multimedia documents usually contain substantial textual parts. Hence a QL for multimedia data should admit the use of *mature text retrieval techniques* and especially the combination of text retrieval techniques with retrieval techniques for other media types. (5) Due to the heterogeneous nature of multimedia applications there is no single combination of different similarity measures fitting well in all application areas. Thus, the QL must facilitate a *flexible combination of different sim-*

*ilarity measures* trimmed well for application specific needs. (6) For a general applicability the QL must not rely on a specific schema or a specific type of data modeling. (7) Last but not least a good performance for all types of queries has to be assured.

In this paper we describe POQL$^{MM}$ as an approach for such a general purpose query language for multimedia databases. POQL$^{MM}$ is an extension of the query language POQL [10, 11] based on the semantically rich data model of PCTE – the ISO and ECMA standard for a public tool interface for an open repository [19]. POQL$^{MM}$ fulfills the requirements mentioned above by means of powerful regular path expressions, several text retrieval techniques including advanced pattern matching and the vector space model, a broad variety of feature extracting operators and similarity measures for similarity searches on multimedia data.

The rest of the paper is organized as follows: In the section 2 we will present an example schema to clarify the requirements for a QL for multimedia data and as a basis for the examples in the following sections. Thereafter, section 3 describes the distinguishing features of POQL$^{MM}$. In section 4 we will discuss, how POQL$^{MM}$ differs from related approaches. Finally, section 5 concludes the paper.

## 2 Conceptual Schema

As pointed out in the introduction, POQL$^{MM}$ is designed as a general purpose query language for multimedia data. Therefore the application of POQL$^{MM}$ does not rely on the use of a specific schema. Nevertheless we will need an example schema in the following sections to describe POQL$^{MM}$ as precisely as possible. To this end, we employ a conceptual schema based on the PCTE data model. It is consciously general, to cover the whole variety of multimedia data. A concrete multimedia application will usually apply a more concise schema.

In order to describe our example schema, we have to give a short introduction into the PCTE data model: The object base contains objects and relationships. Relationships are normally bidirectional. Each relationship is realized by a pair of directed links, which are reverse links of each other. A link type is given by a name, an ordered set of key attributes, a set of non-key attributes, a set of allowed destination object types and a category. For our example schema only two link categories are needed: *composition* (defining the destination object as a component of the origin object) and *reference* (assuring referential integrity).

Now that we have sketched the PCTE data model, we are in the position to describe our example schema, which is shown in figure 1. The attribute types applied to each object type are given in the ovals at the upper left corner of the rectangles representing the object types. The link types representing the relationships between the object types are indicated by arrows. A double arrowhead at the end of a link indicates that the link has cardinality *many*. Links with cardinality many must have a key attribute. In the example the attribute *id* is used for this purpose as long as a numerical key attribute is used. In case of a string attribute we use an attribute named *name*. A *'C'* or *'R'* in the triangles at the center of the line representing a pair of links, indicates that the link in the corresponding direction has category *composition* or *reference*.

The upper right corner of figure 1 represents the model for the document structure. We assume that a multimedia document is made up of one or more substructures called “chunks”, which in turn consist of media objects or lower level “chunks”.

In the lower right corner you can see the representation of the different types of media objects. There are four subtypes (*image*, *video*, *audio*, and *text*) for the object type *media_object* with specific attributes for registration data. The object type *raw_data* is used to store the raw data of a media object in one or more formats.

The part of our schema, depicted in the lower left corner, represents the potential segmentation of media objects. For example an

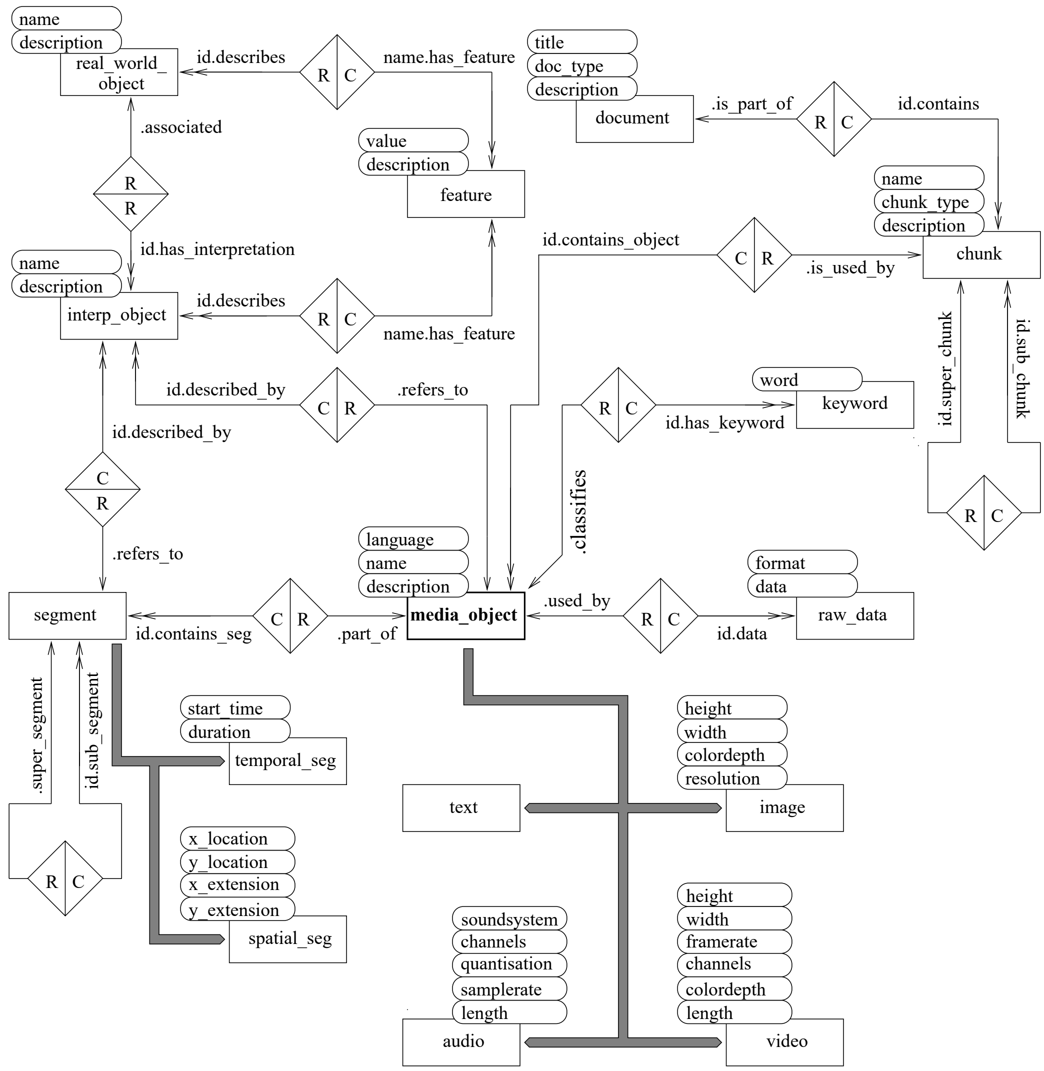


Figure 1: Example schema

image might be segmented into regions containing certain conceptual objects – we call this a spatial segmentation – or a video or an audio might be segmented into shots or songs – we call this a temporal segmentation. The assumption is that the data is segmented manually or automatically (see e.g [14] for an overview on automatic image segmentation algorithms) when it is inserted into the database. The segments are stored in the database as a spatial or temporal "reference" to the original media object. Moreover, segments themselves can contain further subsegments.

As suggested in [1] a "multimedia interpretation model" is added to the schema. This part is given in the upper left corner. It describes the content of media objects and segments by interpretation objects (*interp_object*). For example an image showing Steffi Graf and Andre Agassi might be associated with two interpretation objects, representing their appearance in the image. Note that the interpretation objects can be further linked to conceptual real world objects representing Steffi Graf and Andre Agassi themselves. In addition, features of interpretation objects and real world objects can be maintained by the object type *feature*. Finally the object type *keyword* allows the assignment of keywords to media objects in order to classify them further.

# 3 POQL$^{MM}$

In the following subsections we will focus on the distinguishing features of POQL$^{MM}$. In general POQL$^{MM}$ can be used similar to OQL or even SQL. A query in POQL$^{MM}$ is either a select-statement, or the application of an operator like *count*, *sort+*, *sort–*, or *union*.

## 3.1 Structured Documents

To deal with structured documents suitably, a query language must allow to address subcomponents, super-components and – in general – related components of an entity under consideration in a query in a compact and nevertheless powerful way. In our opinion regular path expressions are extremely useful in this respect, because regular path expressions are a flexible means to define the traversal of links in a query. With POQL$^{MM}$ the basis for regular path expressions are so-called *link definitions*. A link definition can be based either on the link type and the link key attribute values or on the link category. To define the matching key attribute values for string attributes '*' and '?' can be used as wildcards ('*' = zero or more arbitrary characters; '?' = exactly one arbitrary character). For natural key attribute values '_' matches an arbitrary value and in addition exact values, sets and intervals can be defined. For example `_.contains` matches all links of type *contains* irrespective of the actual key attribute value.

When a link definition shall be based on link categories the allowed categories – represented by their first character – have to be given in curly brackets. For example the link definition `{c}` matches all *composition* links. To allow for more flexibility a `shield`-clause can be used to prohibit certain link types or destination object types. For example `{c shield media_object}` matches all *composition* links except those for which the destination object is of type *media_object*.

Link definitions can be concatenated with slashes ('/'). Furthermore POQL$^{MM}$ provides iteration facilities for path expressions. For example `[{c shield media_object}]*` means that zero or more links matching the link definition `{c shield media_object}` have to be traversed. In fact, not only link definitions can be used in the `[...]*` construction but arbitrary path definitions (e.g. consisting of a sequence of link definitions separated by slashes). Moreover POQL$^{MM}$ knows `[` *path definition* `]+` to indicate that a path matching *path definition* has to be traversed at least once and `[` *path definition* `]` to indicate that a path matching *path definition* is optional [1].

As an example for the application of regular path expressions let us consider a query searching for the name and the keywords of all media objects in the document titled "Legends" which have an associated interpretation object referring to the real world object named "Elvis Presley". Note that the corresponding interpretation object can be attached either directly to the media object itself or to a segment of the media object. This might be expressed in POQL$^{MM}$ as follows:

```
select MO:name, MO:_.has_keyword/->word
 from D in document,
     MO in D:[{c}]+/_.contains_object/->.
 where D:title = "Legends"
   and "Elvis Presley"
         in MO:[{c}]+/.associated/->name
```

In this query the document with the title "Legends" is addressed in base set $D$. Base set $MO$ consists of all media objects which can be reached from the document under concern via a path matching the regular path expression `[{c}]+/_.contains_object/->.` . This path expression starts traversing one or more *composition* links. Thereafter a *contains_object* link is traversed pointing to an object of type *media_object*. At the end of the path expression the notation `/->` defines that the destination object is addressed and the final dot means

[1] The semantics of these constructs is as follows: If an object is reached via the same *link definition* (which may be the last link definition in a [ *path_definition* ], [ *path_definition* ]+ or [ *path_definition* ]* for the second time, the investigation of the path under concern is stopped without any further action. This assures termination and brings up the expected result.

that the object itself has to be included into the calculated set. Note that we would use `->` instead of `/->` to address the last link of the path. Summarizing, the regular path expression calculates the set containing all objects which can be reached from the actual object in base set $D$ via a path consisting of one or more *composition* links and a final *contains_object* link.

In the `where`-clause the regular path expression `MO:[{c}]+/ .associated/->name` is used to calculate a set (resp. bag) containing the names of all real world objects connected either to the media object itself or to one of its segments.

Finally the `select`-clause in the example defines that the *name* of the media object and the associated keywords are requested. To calculate the keywords, the *word* attribute values of all *keyword* objects which can be reached via a *has_keyword* link are collected via a regular path expression.

### 3.2 Text Based Retrieval Facilities

Usually structured multimedia documents contain significant textual data like text objects or meta data. Thus POQL$^{MM}$ has integrated text retrieval facilities inherited from POQL comprising pattern matching facilities analogous to the grep command in UNIX and operators implementing the vector space model. Due to space limitations we will only sketch the integration of the vector space model here. For a description of the pattern matching facilities the reader is referred to [11].

The vector space model [21] assumes that an available term set is used to identify both, maintained documents and information requests. Queries and documents are represented as *term vectors* of the form $D_i = (a_{i1}, a_{i2}, \ldots, a_{it})$ and $Q_j = (q_{j1}, q_{j2}, \ldots, q_{jt})$ where $t$ is the number of terms in the term set and where the coefficients $a_{ik}$ and $q_{jk}$ represent the relevance of document $D_i$ or query $Q_j$, respectively, with respect to term $k$. In the literature various term-weighting formulas have been proposed to calculate the $a_{ik}$ and $q_{jk}$. In POQL we employ the formulas presented in [22] which have been proved to be competitive e.g. in [8].

To integrate these formulas into POQL, we use three operators. First there are the unary operators `D_vector` and `Q_vector` which determine a document or query description vector, respectively. Roughly spoken these operators calculate the vector components for the terms based (1) on the term frequency of the terms in the document under concern and (2) on the inverse collection frequency which is high for terms occurring in only a few documents and low for terms occurring in many documents (see [11] for the details). The third operator is the binary operator `sim` calculating the conventional vector product of a document and a query description vector. This yields high values for "similar" documents and low values for "unrelated" documents.

An example for the use of the vector space model in a multimedia context might be to search for a GIF image dealing with *sorting algorithms* and especially *quicksort*. In this case we search for an *image* object for which there is a *raw_data* object with format "GIF". To rank the images fulfilling this condition, we can address the text attributes of the *image* object itself and of all objects which can be reached from this object via *composition* links – according to our schema these objects comprise the associated interpretation information. This is done in the following POQL$^{MM}$ query applying the `D_vector` operator to the "components" of the *image* object determined by the regular path expression "`[{c}]*/->.`". The query determines the 25 *image* objects with the highest relevance for the query text "sorting quicksort":

```
head[25](
sort-(
 select (Q_vector "sorting quicksort"
          sim D_vector I:[{c}]*/->.),
         D:data
 from I in image, D in I:_.data/->.
 where D:format = "GIF"))
```

In this query the query text – which should be longer than two words in practical applications – is given as a string constant. The Q_vector operator is applied to this query text to create the query description vector. The document description vectors are determined applying the D_vector operator to the text in the "components" of the *image* object (recall that *image* is a subtype of *media_object* in our schema) defined by the regular path expression "[{c}]*/->.". This means that the union of all string attributes of all objects contained in the set determined by this regular path expression is considered as the text of the image under concern.

The select-statement yields a multiset of pairs, where the first component contains the similarity value for the actual *"image"* and the second component contains the image in GIF format. The sort- operator is applied to the result of the select-statement yielding a list with the pairs sorted in descending order according to their similarity values. Then the head operator is used to extract the 25 elements at the beginning of this list.

### 3.3 Feature Extraction

The features of POQL$^{MM}$ presented so far can be used to retrieve media objects either addressing the interpretation objects directly or addressing the textual parts of the data by information retrieval techniques. To address the maintained audio, video and image objects respective operators are needed.

We can distinguish three principal types of operators (cf. table 1):

*Conversion* operators accomplish a conversion from one media type to another. For example an OCR operator might yield a text from an image object and a key frame extraction operator might extract a small set of images from a video.

*Segmentation* operators divide a media object into sub-objects of the same type. Examples range from a shot detection operator for videos to region-based or edge-based image segmentation operators. Segmentation is important from the query language point of view, because the resulting sub-objects are usually more specific – and therefore less ambiguous for the specification of query conditions [2].

*Feature extraction* operators represent certain characteristics of media objects in a convenient representation. For example an operator could derive a set of musical instruments from an audio depicting the instruments used in the song production. Other examples are operators for the derivation of a color histogram and the texture of an image.

As an example for *feature extraction* let us search for images with a "mood" similar to an image which is stored in the file "sunrise.gif". Furthermore "sun" should be among the keywords of the image:

```
head[25](
 sort+(
  select (col_hist +.data/->data
           E_dist
          col_hist "FILE:sunrise.gif"), .
  from image
  where "sun" in _.has_keyword/->word))
```

In this query the base set contains all images which are restricted to images with an associated keyword *"sun"* in the where-clause. The select-statement yields a multiset of pairs, where the first component contains the similarity value for the actual *"image"* and the second component contains an object reference for the corresponding *image* object. The unary operator col_hist calculates the color histogram feature vectors from the image data indicated in the operand. Afterwards the similarity of the image histograms is measured with the help of the binary operator E_dist, which simply determines the Euclidean distance between two feature vectors. The sort+ operator is applied to the result of the select-statement yielding a list with the pairs sorted in ascending order according to their distance values. Then the head operator is used to extract the 25 elements at the beginning of this list.

[2] Please note that in our opinion segmentation should be better represented in the schema — as with our example schema. Nevertheless, a general purpose QL for multimedia data has to provide corresponding facilities because of the requirement for schema independence.

| | *feature extraction* | *segmentation* | *conversion* |
|---|---|---|---|
| **audio** | musical instrument detection<br>speaker recognition | interrupt detection | speech recognition |
| **video** | motion detection | shot detection | key frame extraction<br>caption recognition |
| **image** | color histogram<br>texture | region based segmentation<br>edge based segmentation | OCR |

Table 1: Example operators for feature extraction, segmentation, and conversion

### 3.4 Similarity Queries

In conjunction with operators for feature extraction a QL has to provide similarity measures for the extracted feature values, to allow for similarity queries. Examples are the Euclidean distance used in the previous section and the conventional vector product used with the vector space model. However, for the combination of feature extraction and similarity calculation in a QL, two different approaches are conceivable: (1) We could integrate the feature extraction and the similarity calculation in one operator receiving two media objects as operands and calculating a similarity value as its return value. (2) We could use separate operators for feature extraction and similarity calculation — as with the example in the previous section. We prefer the second approach, because it has the advantage of more freedom with the application of similarity measures. In the above query example we might e.g. replace the Euclidean distance with an operator calculating the histogram intersection.

### 3.5 Combining Similarity Measures

The integration of retrieval facilities for different media types into a closed descriptive query language allows to combine the facilities in a flexible way. For example we can combine similarity searches with fact conditions as in the above examples. Furthermore we can combine different sources of evidence for the relevance of an object. E.g. we can combine a similarity search based on text with a similarity search on image features. This can be done in a rather straightforward way multiplying or adding similarity values calculated for the text similarity and the image similarity using the arithmetical operators of POQL$^{\mathrm{MM}}$. Another approach is to merge ranking lists of different sources into a combined ranking list. This problem is known in the literature as “data fusion” [6, 7]. For the merging of the individual lists different strategies are conceivable. In general the problem can be interpreted as a selection problem for which all corresponding techniques from the area of decision theory can be applied. For example we could use one similarity criterion as the dominant one and consider the remaining criteria only when the first criterion yields the same value for a set of objects. An advantage of this approach is that it relieves us from the burden of performing some type of normalization among the different criteria. Nevertheless, the approach seems to be too extreme, because of the dominance of one criterion.

A more flexible approach, which does nevertheless not require a normalization among the criteria, is to use the ranks of the objects with respect to the single criteria. Let $r_{i,j}$ be the rank of object $i$ with respect to criterion $j$ ($j \in \{1, \dots, m\}$ and $r_{i,j} \in \{1, 2, \dots\}$) and let $w_j$ be the weight of criterion $j$ representing its relative importance amongst the criteria. Then we can use the sum $\sum_{j=1}^{m} w_j \cdot \frac{1}{r_{i,j}}$ to derive the combined ranking. Let us consider an example with three criteria and $w_1 = 60$, $w_2 = 20$, and $w_3 = 20$. In this case an object $k$ with $r_{k,1} = 1$, $r_{k,2} = 6$, and $r_{k,3} = 8$ would yield a value of $60 \cdot \frac{1}{1} + 20 \cdot \frac{1}{6} + 20 \cdot \frac{1}{8} = 65,83$. For object $l$ with $r_{l,1} = 2$ the only chance to be ranked ahead of object $k$ in the combined criterion would be to have $r_{k,2} = 1$, and $r_{k,3} = 1$ which yields a combined value of 70. The combination method sketched above is integrated in POQL$^{MM}$ by the `combine`-operator. This operator can be interpreted as a generalized `sort`-operator. The `combine`-operator is applied to tuples. After the `combine` keyword we have to define how the components of the tuples shall influence the sorting. For example `combine[(60, '-'), (20, '+'), (20, '+')]` defines, that the sorting with respect to the first component has to be done in descending order (`'-'`), whereas the sorting with respect to the second and the third component has to be done in ascending order (`'+'`). Furthermore the given numbers represent the values $w_j$.

The following example searches for GIF images dealing with *searching* and *quicksort* which are similar to a sketch given by the user with respect to *color usage* and *texture*.

```
head[25](
 combine[(60,'-'), (20,'+'), (20,'+')](
  select
   (Q_vector "searching quicksort"
    sim D_vector I:.is_used_by/[{c}]*/->.),
   (col_hist "FILE:sketch.gif"
    E_dist col_hist D:data),
   (texture "FILE:sketch.gif"
    E_dist texture D:data)),
   D:data
  from I in image, D in I:_.data/->.
  where D:format = "GIF"))
```

In the example the similarity with respect to the vector space model is given a weight of 60 whereas the color and the texture similarity are each given a weight of 20. The select-statement yields quadruples where the first component represents the textual similarity, the second component represents the color similarity, the third component represents the texture similarity and the fourth component contains the image in GIF format. The `combine`-operator sorts the result based on the first three components as described above.

### 3.6 Schema Independence

It might be a matter of discussion, whether feature extraction, segmentation and conversion should be incorporated into the query language or into the schema. In fact there are good reasons to represent the segmentation of media objects in the schema. For example this allows to link interpretation objects — and in consequence real world objects — to segments instead of complete media objects. However, since POQL$^{MM}$ is designed as a general purpose QL which should not rely on a specific schema or a specific type of modeling of the data, an integration of corresponding facilities into the query language is indispensable. Furthermore an integration of such facilities into the query language allows for more flexibility.

### 3.7 Performance

Last but not least the performance of a QL should not be neglected. In order to achieve a satisfactory performance for complex queries combining fact conditions and similarity based retrieval, specialized index structures are applied with our implementation of POQL$^{MM}$. To this end, we employ an adapted version of our LSD$^h$-tree [12]. When multiple similarity criteria are combined applying the `combine`-operator described in section 3.5, a variant of the algorithm presented by Pfeifer and Pennekamp [20] is used. Roughly spoken this algorithm performs parallel similarity searches on different access structures until the top positions in the resulting list are stable. The next considered element in the performed loop is always taken from the structure for which the next element can contribute the highest amount to the overall similarity. In order not to calculate the feature vectors at query time, the feature vectors of media objects are determined and stored in the index structure when a document is inserted into the database.

## 4 Related Work

The work done in four related research areas has stimulated our work, namely (1) query languages for multimedia data (2) path expressions to simplify navigation in object-oriented queries (3) multimedia database systems and multimedia information retrieval and (4) the treatment of structured documents.

Within the research area of multimedia query languages a lot of papers have been published in the last years. The approach most similar to our query language is MOQL (Multimedia Object Query Language) [17]. MOQL is a general multimedia query language based on ODMG's Object Query Language (OQL). Furthermore a visual front end for MOQL exists. However, MOQL mainly addresses the complex spatial and temporal relationships inherent in a wide range of multimedia data types. In contrast, our language aims mainly for structural and feature-based best-match and similarity queries.

As described in section 3.1 POQL$^{\mathrm{MM}}$ employs regular path expressions to compute sets of objects or sets of attribute values which can be reached via links in a query. In recent years many papers dealing with path expressions in object oriented databases have been published (see e.g. [9, 15]). In contrast to POQL$^{\mathrm{MM}}$ most of these papers use path expressions as some kind of abbreviation to specify a desired connection between two objects. Hence, some of them discuss several strategies which can be used when the connections are ambiguous. Instead, we use path expressions to define a set of objects.

The research area of multimedia database systems and multimedia information retrieval has been rather active in recent years (see e.g. [2, 23] for an overview on the work done in these areas). However, various of these approaches are either dedicated to a single media type like images [18, 5], strongly application specific [24], or high level approaches [1]. Nevertheless many ideas from this area influenced our approach.

Another related area is the management of structured documents. In [3] for example a mapping from SGML documents into OODBs and an extension of an SQL-like OODB query language is given in order to deal with SGML document retrieval. Another example is W3QS described in [16]. W3QS is a query system for the World-Wide Web and includes an SQL-like query language providing pattern matching facilities and graph patterns which can be used to address the structure of WWW documents. In [4] HyperFile is presented as a data and query model for hypertext documents. HyperFile can be seen as an add-on system that enhances a DBMS for hypertext management. However, these approaches hardly address similarity based queries for text and other media types.

# 5 Conclusion

In this paper we have proposed POQL$^{\mathrm{MM}}$ as a general purpose query language for multimedia databases. Although POQL$^{\mathrm{MM}}$ is based on the data model of PCTE most of the presented facilities are by no means restricted to this data model. The main distinguishing features of POQL$^{\mathrm{MM}}$ are (1) the incorporation of powerful regular path expressions to deal with structured documents, (2) the integration of various feature extraction operators for the conceivable media types and (3) the `combine`-operator allowing for a flexible combination of different similarity criteria. Future research directions for POQL$^{\mathrm{MM}}$ include the development of application specific end-user interfaces on top of POQL$^{\mathrm{MM}}$ to validate our approach. Furthermore, a sophisticated query optimizer taking into account the regular path expressions – and their potential invertability – as well as combined similarity searches is an interesting issue.

# References

[1] A. Analyti and S. Christodoulakis. *Content-Based Querying*, chapter 8, pages 145–180. In Apers et al. [2], 1997.

[2] P. M. G. Apers, H. M. Blanken, and M. A. Houtsma, editors. *Multimedia Databases in Perspective.* Springer, London, 1997.

[3] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pages 313–324, Minneapolis, Minn., 1994.

[4] C. Clifton, H. Garcia-Molina, and D. Bloom. Hyperfile: A data and query model for documents. *VLDB Journal*, 4(1):45–86, 1995.

[5] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos. Target testing and the pichunter bayesian multimedia retrieval system. In *Proc. 3rd Forum on Research and Technology Advances in Digital Library*, pages 66–75, Washington, D.C. USA, 1996. IEEE.

[6] R. Fagin. Combining fuzzy information from multiple systems. In *Proc. 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June, 1996, Montreal, Canada*, pages 216–226, 1996.

[7] U. Güntzer, W.-T. Balke, and W. Kießling. Optimizing multi-feature queries for image databases. In *VLDB 2000, Proc. 26th Intl. Conf. on Very Large Data Bases*, pages 419–428, Cairo, Egypt, Sept. 2000.

[8] D. Harman. Overview of the second text retrieval conf. (TREC-2). *Information Processing and Management*, 31(3):271–289, 1995.

[9] C. Harrison. An Adaptive Query Language for Object-Oriented Databases: Automatic Navigation Through Partially Specified Data Structures. Technical Report NU-CCS-94-19, Northeastern University College of Computer Science, 1994.

[10] A. Henrich. P-OQL: an OQL-oriented query language for PCTE. In *Proc. 7th Conf. on Software Engineering Environments (SEE '95)*, pages 48–60, Noordwijkerhout, Niederlande, 1995. IEEE.

[11] A. Henrich. Document retrieval facilities for repository-based system development environments. In *Proc. 19th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 101–109, Zürich, 1996.

[12] A. Henrich. The LSD$^{\mathrm{h}}$-tree: An access structure for feature vectors. In *Proc. 14th Intl. Conf. on Data Engineering, February, 1998, Orlando, Florida*, pages 362–369. IEEE Computer Society, 1998.

[13] A. Henrich and G. Robbert. Combining multimedia retrieval and text retrieval to search structured documents in digital libraries. In *Proc. 1st DELOS Workshop on Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, December 2000.

[14] B. Jähne. *Digital Image Processing. Concepts, Algorithms, and Scientific Applications.* Springer-Verlag, Berlin, 1997.

[15] M. Kifer, W. Kim, and Y. Sagiv. Querying Object-Oriented Databases. In *Proc. ACM SIGMOD Annual Conf. on Management of Data*, pages 393–402, San Diego, Cal., 1992.

[16] D. Konopnicki and O. Shmueli. W3QS: A query system for the world-wide web. In *Proc. 21th Intl. Conf. on Very Large Data Bases (VLDB '95)*, pages 54–65, Zürich, 1995.

[17] J. Z. Li, M. T. Özsu, D. Szafron, and V. Oria. Moql: A multimedia object query language. In *The Third Intl. Workshop on Multimedia Information Systems*, pages 19–28, Como, Italy, 1997.

[18] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: Querying images by content, using color, texture, and shape. In *SPIE Proc. Vol. 1908*, pages 173–187, San Jose, 1993.

[19] Portable Common Tool Environment - Abstract Specification / C Bindings / Ada Bindings. ISO IS 13719-1/-2/-3, 1994.

[20] U. Pfeifer and S. Pennekamp. Incremental Processing of Vague Queries in Interactive Retrieval Systems. In *Hypertext - Information Retrieval - Multimedia '97: Theorien, Modelle und Implementierungen integrierter elektronischer Informationssysteme*, pages 223–235, Dortmund, 1997. Univ.-Verlag Konstanz.

[21] G. Salton. *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, Mass., 1989.

[22] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[23] V. S. Subrahmanian and S. Jajodia, editors. *Multimedia Database Systems – Issues and Reasearch Directions.* Springer, Berlin, Heidelberg, 1996.

[24] J.-K. Wu, B. M. Mehtre, Y. J. Gao, C.-P. Lam, and A. D. Narasimhalu. STAR—A multimedia database system for trademark registration. In *Applications of Databases, 1st Intl. Conf.*, volume 819 of *LNiCS*, pages 109–122, Vadstena, Sweden, 1994. Springer.