Technical Report - Ulmer Informatik-Berichte No. 99-06, University of Ulm, August, 1999

Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

Utz Westermann, Wolfgang Klas
Databases and Information Systems (DBIS), Computer Science Department
Oberer Eselsberg, University of Ulm
James-Franck-Ring, 89069 Ulm, Germany
Tel: +49-731-50-24131, Fax:+49-731-50-24134
{klas, westermann}@informatik.uni-ulm.de

Abstract

Advanced multimedia applications require adequate support by database technology for the integrated, uniform management, retrieval, and delivery of media data of different types. While there has been substantial effort to provide database support for multimedia data and for its content-based retrieval, research has mostly focused on support for single media types, usually videos or images. However, this has led to a variety of isolated solutions of database support optimized for the respective media data types which are difficult to integrate — a uniform view is lacking. In this paper, we describe the design and the architecture of the Media Integration Blade, a DataBlade module for the object-relational DBMS Informix Dynamic Server/ Universal Data Option. Building upon existing media type-specific DataBlade modules, the Media Integration Blade establishes an integration layer which offers uniform, homogeneous access to the different types of media data. It allows for the uniform retrieval of media data of different type by technical characteristics and content while maintaining the functionality of the underlying media type-specific DataBlade modules. Hereby, the Media Integration Blade forms a generic core component that can be employed for the uniform management and access to media data of different types in any database-driven multimedia information system.

1 Introduction

The advantages of building multimedia information systems on top of database systems have long been recognized [KA97, RNL95, AN97]. The nature of multimedia data, however, imposes requirements on database systems. In general, media data have high volume and, in case of continuous media data types, are time-dependent. To provide for the management and content-based retrieval of media data, support for media data types must be seamlessly integrated with the database management system (DBMS) [KA97].

So far, research in the area of multimedia database systems has focused on database support for specific single media data types, mostly video and image. This results in a variety of available prototypes. For example, the Surfimage system [NMB$^+$98] and the QBIC system [F$^+$95] are image database prototypes which allow to query their contents by example images using sophisticated similarity search algorithms. The STARCH database [BG99] facilitates querying for images with the help of an ontology specified by description logics. As an example of a video database, the OVID system [OT93] allows to define sets of consecutive frames as video objects that can be queried by the VideoSQL query language. Jiang and Elmagarmid [JE98] introduce a prototype of a video database based on the Logical Hypervideo Data Model that allows to define video objects inside video frames which can be indexed by free text and connected to other video objects via hyperlinks. In literature, one can also find some few examples of databases concerned with the management of audio data and speech [LR95, WS98]. On the commercial market, database technology is evolving that supports different media data types. For instance, the object-relational DBMS Informix Dynamic Server/ Universal Data Option (IDS/UD) can be extended by DataBlade modules for support of media data like image, text, and video.

As the prototype systems and database extensions concentrate on just one particular media data type, they vary in the location where media data is stored, in the way technical metadata is available, and in the way content-based retrieval of media data is performed. Considering the storage location of media data, variants range from storage of media data in binary large objects inside a database, in a file system, on a web server, or even on a full-fledged media server allowing for the streaming delivery of continuous media data. Handling of technical metadata is focused on each specific media data type. In consequence, for some media data types the support for technical metadata is not necessarily comprehensive. For instance, the Informix Video Foundation DataBlade offers no support for color depth at all, though color depth is desirable for videos. Additionally, technical metadata applicable to different media data types, e.g., the size of image and video, may be handled and named differently. Content-based retrieval of media data ranges from similarity search based on automatically extractable features like the color distribution of an image to search based on content-descriptive semantic annotations.

As long as an application is concerned with only a single media data type, these variations cause no problems. There are, however, applications that need to manage media data of different types in a uniform and integrated fashion. To overcome the variations, such an application itself has to cope with the heterogeneity of the media data type-specific concepts. For instance, depending on the storage location of a medium and its particular media data type, the application has to employ different mechanisms to access the media data. Moreover, content-based retrieval of media data spanning several different media types is difficult to achieve by the application, as it must map a given query to several media type-dependent facilities for content-based retrieval. Obviously, it is not possible to apply highly effective similarity search techniques based on the color distribution of images to audios. Thus, means for content-based retrieval applicable to different media data types is a necessity. Finally, it is difficult for the application to limit the search for media data using technical characteristics applicable to different media data types such as the size in bytes since media data type-specific prototype systems and database extensions manage technical metadata differently. Here, an integrated view on media data would be helpful.

With the project "Gallery of Cardiac Surgery" (Cardio-OP[1])[KGF99], that aims at the development of an Internet-based and database-driven multimedia information system in the domain of cardiac surgery, we find such an application that explicitly requires uniform management and uniform content-based retrieval of multimedia material of different types ranging from videos, images, and texts, to full-fledged multimedia presentations. Based on a multimedia repository, the system is going to serve as a common information and education base for its different types of users, physicians, medical lecturers, students, and patients, who are provided with multimedia data according to their user specific request to the multimedia information system, their different understanding of the selected subject, their location and technical infrastructure. The underlying database technology of the repository is given by the object-relational DBMS IDS/UD which has been chosen for reasons of flexibility, profound extensibility, and industrial strength [BKW99a].

In this paper, we describe the design and the architecture of the Media Integration Blade (MIB) DataBlade module for IDS/UD which establishes an integration layer upon media data type-specific DataBlade modules to overcome the heterogeneity of the media data type-specific concepts and to offer applications transparent and uniform access to media data of different media data types. Based on commercial DataBlade modules for text, image, and video data, the MIB provides transparent access to media data of different type by encapsulating the storage location in an abstract data type. The MIB builds an information layer that offers a homogeneous view on the technical metadata of the different media data types. To support uniform, media data type-independent content-based retrieval, the MIB allows for the semantic annotation of media data of different type with concepts taken from a common ontology and by offering query operators for these annotations. However, media data type-specific functionality for content-based retrieval based on automatically extractable features offered by the underlying DataBlade modules remains accessible. The concepts illustrated in this paper and implemented by the MIB are generic to the extent that the overall approach can be applied to other database technologies as well.

The paper is organized as follows: Section 2 shows how to achieve transparent media access. Section 3 describes the organization of the media data to accomplish a homogeneous, integrated view to the technical metadata. Section 4 introduces the annotation and content-based querying facilities of the MIB. Section 5 illustrates the application of the MIB module. Section 6 concludes the paper and gives an outlook to ongoing and future work.

2 Transparent media access

As mentioned in the introduction, the location where media data is stored and accessed is often different in media data type-specific database extensions. Among others, media data may be stored on a media server, on a web server, in a file system, or in binary large objects of a database. Each storage location comes with its own access methods. For instance, access to files is done with the help of operating system routines and a media server might define a specific network protocol to access media data. This heterogeneity of access to media data hinders applications requiring a uniform model of media access. A known solution to this problem is the employment of *locators*. A locator is a data type which abstracts from the way media data is stored by encapsulating its storage location.

---

[1] Cardio-OP - Gallery of Cardiac Surgery - is partially funded by the German Ministry of Research and Education, grant number 08C58456. Our project partners are the University Hospital of Ulm, Dept. of Cardiac Surgery and Dept. of Cardiology, the University Hospital of Heidelberg, Dept. of Cardiac Surgery, an associated Rehabilitation Hospital, the publishers Barth-Verlag and dpunkt-Verlag, Heidelberg, FAW Ulm, and ENTEC GmbH, St. Augustin. For details see also URL www.informatik.uni-ulm.de/dbis/Cardio-OP/

A locator comes with a set of access routines. Depending on the particular storage location encapsulated in an instance of the locator, the access routines use the access methods appropriate for that storage location. Thus, it is hidden from an application where media data is stored and by which method exactly it is accessed as an application always calls the same access routines.

Ideally, the problem of offering transparent access to a variety of storage locations could be solved by employing one comprehensive locator data type. As a matter of fact, different locator types have evolved that, unfortunately, support different storage locations and come with different access routines. For this reason, it is difficult to decide for one of these variants. To illustrate this point, consider the locator variants provided with the Excalibur Text DataBlade, Excalibur Image DataBlade, and the Informix Video Foundation DataBlade modules for the IDS/UD. In particular, the Excalibur Text DataBlade module makes use of the `LLD_Locator` data type. This locator allows for the transparent access to media data stored in the file system of the IDS/UD server or in a binary large object of a database. The Excalibur Image DataBlade module references media data by the use of the data type `IfdLocator` which is essentially similar to `LLD_Locator` but can store additional application-specific information. While the differences between `LLD_Locator` and `IfdLocator` are mainly structural, the locator `MedLoc` introduced by the Video Foundation DataBlade supports other storage locations. The `MedLoc` locator serves to reference media data stored on a media server with the help of the Virtual Storage Interface (VSI), an interface provided by Informix for accessing data on media servers. The Video Foundation DataBlade module provides an implementation of VSI which simply maps media data to files on the file system. Third party vendors can connect their media servers to IDS/UD via `MedLoc` by providing their own implementation of VSI.

For the design of the MIB that intends the integration of the DataBlade modules mentioned above, a suitable concept for a uniform locator has to be developed. This design must take into consideration that the functionality of the underlying media data type-specific DataBlade modules should remain usable for applications. For instance, one would still like to use the functions and index structures of the Excalibur Image DataBlade responsible for the similarity search among images. These functions and data structures expect the media data type-specific locator `IfdLocator`. This locator, however, is not necessarily applicable to functions of other media data type-specific DataBlade modules. Thus, it is neither possible for the MIB to simply decide for one of the locator variants described above nor to simply stay with the variety of locator variants as this contradicts the objective of the MIB to provide uniform access to media data. Therefore, the MIB defines a uniform locator named `mediaLocator` that abstracts from the different variants. It is able to encapsulate an instance of one of the various locator variants. Depending on the encapsulated locator variant, an instance of `mediaLocator` accommodates a suitable structure to represent the type and the data of the encapsulated variant. In order to obtain instances of `mediaLocator`, typecasts from the different locator variants to `mediaLocator` are defined. Typecasts are also defined in the opposite direction to allow for the access of media data type-specific functionality of the underlying DataBlade modules. However, not all instances of `mediaLocator` can be casted to all locator variants. This depends on the locator variant from which the `mediaLocator` instance was constructed. Consider, for example, an instance of `mediaLocator` constructed from the locator variant `MedLoc` that references media data stored on a media server. To cast this instance to the type `LLD_Locator`, the media data would have to be copied to either the file system of the database server or to a binary large object in a database as an instance of `LLD_Locator` can only reference these storage locations. Regarding the high volume of media data, this solution is not feasible. Hence, such typecasts are not supported.

The uniform locator `mediaLocator` has associated access routines that facilitate the uniform, transparent access to media data. For this purpose, the MIB implements several user-defined routines. The routine `mediaLocatorToClient` copies the media data to the file system of the client calling this routine. For access at a finer granularity, the MIB implements routines like `open`, `close`, `read`, and `seek` with the usual file access semantics which in turn make use of the appropriate access methods provided with the specific locator variant encapsulated by the instance of `mediaLocator`.

The locator mechanism of the MIB as described above can be extended to cover additional storage locations for media data with relatively little effort. For this purpose, the type `mediaLocator` must be extended to accommodate a suitable structure to reference media data in the new storage location. Furthermore, the user-defined routines for uniform media access must be extended to integrate the access to the new storage location. So far, we have provided `medLocator` with support for the encapsulation of instances of the locator variants `LLD_Locator`, `IfdLocator`, and `MedLoc` thereby being able to transparently access media data on the filesystem of the database server, in binary large objects in a database, and on media servers supporting VSI. We plan to extend the locator mechanism with the capability to access media data stored on a web server using HTTP.

3 Media organization

In the previous section, we have introduced the concepts provided by the MIB to transparently access media data stored in various locations. In addition to transparent access, an organization of media data is of importance that allows a user to find and select relevant material efficiently. Regarding Cardio-OP, this is on the one hand applies to users looking for information; if they cannot find the information they want quickly, the acceptance of the system will diminish. On the other hand, efficient retrieval and selection of media is also important during an multimedia authoring process; if it is more difficult for an author of multimedia content to find existing media data and to reuse it than to reproduce it, the degree of reuse will be very low which is not cost-effective. Hence, a multimedia repository should support sophisticated, fine-grained retrieval of media data according to media type, the associated technical metadata, and content. For instance, it should be possible to limit a search for media data to those images coded in JPEG format not exceeding a width of 500 pixels and a height of 250 pixels. Additionally, the multimedia repository should support the selection of media data by the information content, or even better, the mixed selection of media data by technical metadata and content. This section concentrates on the organization of media data according to media data type and technical metadata while the support for content-based retrieval is presented in Section 4.

The underlying DataBlade modules, on which the MIB is based, provide technical metadata for the media data types they manage. However, this metadata is spread over the various modules and, as mentioned before, cannot be accessed in an integrated fashion by applications. Metadata applicable to several media data types might be named differently, might follow different units of measurement, or might not be even managed at all by the respective DataBlade modules. What is needed for applications is an integrated view on media data and the associated technical characteristics. Such a view should not restrict itself on technical metadata applicable to all supported media data types. The lowest common denominator would not be very helpful for applications as, apart from the size of media data in bytes and its coding format, there would not be much technical metadata applicable to all media data types.

The approach taken by the MIB is to create a new layer of information that provides a homogeneous and uniform model of media data and the associated technical meta data. From all technical characteristics available with the different media data types, a comprehensive set of features is selected and integrated in this layer. This selection, however, is influenced by the applications that are to be supported. Each technical characteristic of this set is associated with all the media data types it applies to. To support sophisticated cross-media type queries, a further organization of this set is reasonable. A user might be interested in media data which can be viewed but does not care whether it is of type image or video. Another user might not be interested in a limitation to certain media types at all. To tackle this problem, the selected technical characteristics are organized in a specialization hierarchy (see Figure 1). The top of the hierarchy models those technical characteristics applicable to all media data types, i.e. the lowest common denominator. The leaves of the specialization hierachy represent the technical metadata that are applicable to specific media data types. The inner nodes of the specialization hierarchy bring together technical media data of closely interrelated media data types. For example, `Viewable` subsumes technical metadata common to all media data types that can be viewed by humans like width and height. This specialization hierarchy is based on the assumption that the inner nodes reflect distinctions between media data types that are likely to be important for user queries.

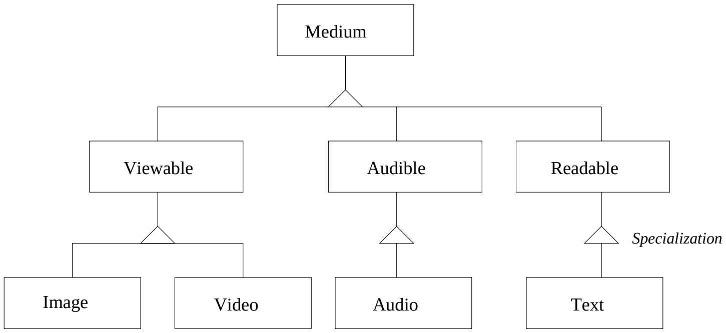


Figure 1: Media organization

Employing this organization, an application is able to search for media data with regard to certain technical characteristics without necessarily limiting the search on one particular media data type. The organization of media data is not limited to the specialization hierarchy as depicted in Figure 1. New media data types can be included in this organization by placing them into the specialization hierarchy under the most suitable inner node. Hereby, the set of technical metadata is extended by additional technical characteristics applicable only to the new media data type. New types of queries can be supported by the reorganization and/or more fine-grained specialization of the hierarchy.

The specialization hierarchy of Figure 1 is realized by the MIB with a table hierarchy, with each table representing a node. Hereby, the MIB utilizes primitives of the object-relational DBMS IDS/UD which allow to define specialization relationships between tables. The columns of these tables model the technical metadata associated with the respective node. An additional column is provided with the root table that references the actual media data with the help of the `mediaLocator`. This establishes the link between technical metadata and media data.

4 Content-based retrieval

In the previous section, we have explained how the MIB organizes media data according to technical metadata. Furthermore, support for content-based retrieval of media data in a uniform fashion, independent of the particular media data type is necessary. However, the facilities for content-based retrieval offered by the media data type-specific database extensions are coined to the respective media data types. Thus, uniform and integrated cross-media type content-based retrieval is difficult to achieve. For the integrated, uniform content-based access to media data of different types, a uniform layer of content-descriptive information is established on which uniform content-based queries can be executed. Therefore, we first introduce the concept of annotations which relates media data of different types to terms taken from a controlled vocabulary organized in a concept hierarchy. Based on this annotation scheme, we informally introduce the different semantics of content-based retrieval to be supported. Following this query semantics, we define a set of query operators for content-based retrieval that implement the introduced semantics.

4.1 Media annotation

Media *annotations* relate media data to *concepts* from an application domain in order to semantically describe media data's content. A concept is an abstract idea of an entity important for a particular application domain. An example of a concept taken from the application domain of Cardio-OP is “chest of the human body”. Each concept is associated with one or more not necessarily unambiguous terms called *captions* in some language. For instance, the captions “thorac” and “chest” are two terms for the concept “chest of the human body”. Concepts are organized in a specialization hierarchy. This means that if media data is related to a concept c_1 which is a subconcept of c_2 then the media data related to c_1 is considered to be related to c_2 as well. The concept hierarchy reflects the knowledge of the application domain.

We distinguish between *discrete annotations* and *continuous annotations*. Discrete annotations relate media data as a whole to concepts. However, in case of continuous media data types, it might be of interest to relate concepts only to temporal intervals of media data rather than to the entire media data. Thereby, a more fine-grained description of content can be achieved. The symbols introduced in Definition 1 are used in the formal definitions to follow. In Definition 2 we then introduce the notions of discrete annotation and continuous annotation formally.

Definition 1 — Symbols: M**,** CM**,** DM**,** C**,** T**,** $duration$**,** I_m

Let M denote the set of all media data.

Let $CM \subseteq M$ denote the set of all media data of continuous media data type.

Let $DM \subseteq M$ denote the set of all media data of discrete media data type.

Let C denote the set of concepts organized in the concept hierarchy.

Let T denote the set of all captions associated with the concepts C.

The function $duration$ returns the duration of a continuous medium $m \in CM$.

For all $m \in CM$, let I_m denote the set of all valid time intervals i of m, $i = [s, e]$, with $s \leq e$, $s \geq 0$, and $e \leq duration(m)$. ☐

Definition 2 — Annotation

The triple $a_m = (m, c, i)$, $m \in M$, $c \in C$, $i \in I_m \cup \{\varepsilon\}$ is called an *annotation* of m. If $i = \varepsilon$ then a_m is called a *discrete* annotation. Otherwise, a_m is called a *continuous* annotation. The following condition must hold for a_m: $m \in DM \Rightarrow i = \varepsilon$.

The annotation $a_m = (m, c, i)$ is called *suiting to caption* $t \in T$ iff t is a caption associated with c or a subconcept of c. ☐

Definition 2 ensures that only continuous media data can be annotated continuously. With the notion of *suiting*, we provide means to find those concepts corresponding to a given caption.

For the management of the concept hierarchy, the MIB relies on the commercially available COCOON DataBlade module by dimedis [dim98] . The MIB models annotations by providing a table called `Annotations`, each row representing an annotation. This table features one column for referencing the media data, one column which contains the id of the concept with which the media data is annotated, and a column describing the time interval in which the annotation is valid. This latter column is `NULL` in case of a discrete annotation. The table `Annotation` realizes the association of concepts with media data.

4.2 Semantics of content-based retrieval

The annotations introduced in the previous subsection offer a means to describe the content of media data independent of the respective media data type. In order to exploit these annotations for content-based retrieval, we distinguish between *continuous* and *discrete* query semantics.

In the discrete query semantics, annotations of both discrete and continuous media data are apprehended as related to the entire media data. Continuous annotations are treated as if they were discrete, i.e., $a_m = (m, c, i)$, $i \neq \varepsilon$ is considered equivalent to $a'_m = (m, c, \varepsilon)$. The results of queries with discrete semantics are alway references to entire media. Consider, for example, the annotated video v shown in Figure 2. With discrete semantics, a query for media data referring to concepts “thorac” and “infusion” would return a reference the entire video v, even though these concepts have been annotated only to intervals a and b of v, respectively.

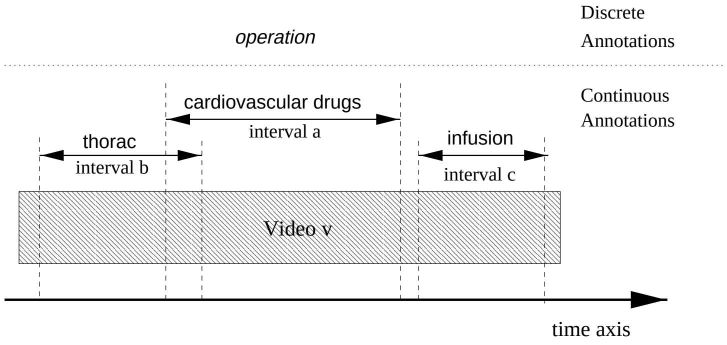


Figure 2: Video v with discrete and continuous annotations

The granularity of the results of queries with discrete semantics is very coarse-grained especially if continuous media data are involved. It is not very helpful for a user to retrieve a video clip with a duration of one hour that at some point in time relates to certain concepts. The user would have to manually search through the video clip to find the time intervals with the relevant content. A user would rather like to know *when* the content refers to the desired concepts. For this kind of queries, we provide continuous query semantics. In contrast to the discrete query semantics, the continuous semantics respect the time intervals of continuous annotations.

In case of discrete media data, the continuous query semantics return references to entire media data just as with discrete semantics. In the case of continuous media data, however, the continuous query semantics return references to time intervals of continuous media data. These are the time intervals of continuous media data that are relevant with regard to the query. Discrete annotations of continuous media data are treated as continuous annotations that apply to the entire media data, i.e., the discrete annotation $a_m = (m, c, \varepsilon)$ is considered equivalent to the continuous annotation $a'_m = (m, c, [0, duration(m)])$. Consider a query for media data referring to the concepts “thorac” *and* “infusion” with continuous semantics. Regarding the example given in Figure 2, it is the question whether the depicted video v would qualify or not as a result for this query. If it was sufficient that at least two intervals exist in v each associated with one of the concepts, v would be part of the query result. However, if the qualifying condition additionally demands that the intervals which bear relevant annotations for the query must overlap, v is not relevant to the query.

In order to offer a comfortable means for content-based access to media data, query operators for the conjunction, disjunction, and negation of concepts should be provided with an exact semantics. As we have seen in the examples above, these operators differ depending on the query semantics. Thus, the operators should be formally defined for each query semantics which is done in the following subsection.

4.3 Query operators

For the two different query semantics, we now formally define the query operators to realize these semantics. These query operators are $retrieve$, and, or, and not, each of which is provided for both discrete and continuous query semantics. These operators return sets of references to media data as a result. Like annotations, such a reference may be *discrete*, in case the whole media data is referenced, or *continuous*, in case only a time interval of media data of a continuous media data type is referenced. Definition 3 formally introduces the notion of references.

Definition 3 — Reference

The tuple $r_m = (m, i)$, $m \in M$, $i \in I_m \cup \{\varepsilon\}$ is called a *reference* to m. If $i = \varepsilon$ then r_m is called a *discrete* reference. Otherwise, r_m is called a *continuous* reference. The following condition must hold for r_m: $m \in DM \Rightarrow i = \varepsilon$. ☐

Given the notion of reference, we now can introduce the first of the query operators, $retrieve$, in Definition 5. The operator allows to select the media data referring to concepts described by a given caption $t \in T$. This operator is the cornerstone of the content-based retrieval facilities. The variant for discrete query semantics, $retrieve_d$, returns only discrete references to media data while the variant for continuous query semantics, $retrieve_c$, returns fine-grained continuous references whereever possible. The continuous variant $retrieve_c$ ensures that only continuous references to continuous media data are returned in the result set by converting discrete references on continuous media data m to the equivalent continuous reference on interval $[0, duration(m)]$. This is done to simplify the following formal definitions of query operators for continuous semantics. Additional symbols used in the ensuing definitions are given by Definition 4.

Definition 4 — Symbols: R, A

Let R denote the set of all possible references.

Let A denote the set of annotations. ☐

Definition 5 — Query operator $retrieve$

The query operator for discrete query semantics $retrieve_d : T \to 2^R$ is defined as follows:
$r_m = (m, \varepsilon) \in retrieve_d(t)$ iff $\exists\, a_m = (m, c, i) \in A$ with a_m *suiting to* t.

The query operator for continuous semantics $retrieve_c : T \to 2^R$ is defined as follows:
$\forall a_m = (m, c, i) \in A$: if $i = \varepsilon$ and $m \in CM$ and a_m is *suiting to* t then $r_m = (m, [0, duration(m)]) \in retrieve_c(t)$. Otherwise, if c is *suiting to* t then $r_m = (m, i) \in retrieve_c(t)$. ☐

Considering the video v in Figure 2, the query $retrieve_d$(*"thorac"*) returns the discrete reference (v, ε) among its results. In contrast, the query $retrieve_c$(*"thorac"*) returns the continuous reference (v, b) among its results. The query result for $retrieve_c$(*"operation"*) would encompass the reference $(v, [0, duration(v)])$. This is due to the handling of discrete references on continuous media data mentioned above.

It is not very satisfying to be able to query for the content of media data according to one caption only. Rather, a user might want to use a conjunction of several concepts to query media data. For this purpose, the query operator and is provided which is formally described in Definition 6. As already mentioned above, the discrete query semantics treats discrete and continuous annotations equally. The semantics of the discrete variant of the conjunction and_d is that each media data in the result set is annotated, either discretely or continuously, with all of the desired concepts. The continuous query semantics, however, relate the conjunction not only to the entire media data but to specific intervals; not only the media data are annotated with the desired concepts but also within the same temporal intervals.

It is an objective of our design, that the query operators and, or, and not can be arbitrarily nested, as long the variants for the two different query semantics are not mixed. This allows for the flexible composition of complex queries. To achieve this goal, the operators and, or, and not do not take captions as arguments like the $retrieve$ operator, but rather sets of references. These sets may, of course, be obtained as a result of a $retrieve$ operator. For instance, in order to achieve a conjunction between two concepts, the and operator is employed on two $retrieve$ operators which in turn deliver the references to the media data referring to the desired concepts.

Definition 6 — Query operator and

The query operator for discrete query semantics $and_d : 2^R \times 2^R \rightarrow 2^R$ is defined as follows:
$r_m = (m, \varepsilon) \in and_d(R_1, R_2)$ iff $r_m \in R_1$ and $r_m \in R_2$.

The query operator for continuous query semantics $and_c : 2^R \times 2^R \rightarrow 2^R$ is defined as follows:
$\forall r_m = (m, i) \in R : r_m \in and_c(R_1, R_2)$ iff one of the ensuing conditions holds:

1. $\exists\ r'_m = (m, i_1) \in R_1$ with $i_1 \neq \varepsilon$ and $\exists\ r''_m = (m, i_2) \in R_2$ with $i_2 \neq \varepsilon$, i_1 temporally overlaps i_2 and $i = i_1 \cap i_2$.
2. $\exists\ r'_m = (m, \varepsilon) \in R_1$ and $\exists\ r''_m = (m, i) \in R_2$.
3. $\exists\ r''_m = (m, \varepsilon) \in R_2$ and $\exists\ r'_m = (m, i) \in R_1$.

□

The variant for discrete query semantics and_d basically calculates the intersection between two sets of discrete references. The variant for continuous query semantics and_c calculates the intersection between two sets of references as well, but for this additionally considers the temporal intervals of the continuous references. If the intervals of two continuous references in the two sets to the same medium overlap, then the result of and_c contains a continuous reference to the medium with an interval representing the intersection of the two intervals (condition 1.). The result of and_c applied to two sets containing a discrete and a continuous reference to the same medium always contains the continuous reference to that medium (conditions 2. and 3.). Conditions 2. and 3. also subsume the intersection of two discrete references to the same medium, which results in a discrete reference.

Regarding the video v shown in Figure 2, the query with discrete query semantics $and_d(retrieve_d($*"thorac"*$),$ $retrieve_d$ $($*"infusion"*$))$ returns the reference (v, ε) among its results. However, the same query with continuous semantics $and_c(retrieve_c($*"thorac"*$), retrieve_c($*"infusion"*$))$ does not return any reference to v as the intervals b and c do not overlap. The query $and_c(retrieve_c($*"thorac"*$), retrieve_c($*"cardiovascular drugs"*$))$ returns the continuous reference $(v, a \cap b)$ among its results as the intervals a and b overlap. The result of query $and_c(retrieve_c($*"operation"*$), retrieve_c($*"cardiovascular drugs"*$))$ includes the reference (v, a) as the annotation "operation" is discrete.

Similar to the and query operator that allows for the conjunction of several captions in a query, the or query operator has been provided to support the disjunction of several captions as well. Definition 7 formally introduces the variants for both query semantics of the or query operator.

Definition 7 — Query operator or
The query operator for discrete query semantics $or_d : 2^R \times 2^R \rightarrow 2^R$ as well as the query operator $or_c : 2^R \times 2^R \rightarrow 2^R$ for continuous semantics are defined as follows:
$or_d(R_1, R_2) = R_1 \cup R_2$ and $or_c(R_1, R_2) = R_1 \cup R_2$. □

Following the definition above, the or operator simply calculates the union between two sets of references for both query semantics. Thus, regarding video v of Figure 2, $or_d(retrieve_d($*"thorac"*$), retrieve_d$ $($*"infusion"*$))$ returns among the result set the discrete reference (v, ε). The query $or_c(retrieve_c($*"thorac"*$), retrieve_c$ $($*"infusion"*$))$ returns the continuous references (v, b) and (v, c) among the results.

Finally, we offer the query operator not for negation. It allows to select media data *not* annotated with a given set of concepts. The operator variant for discrete query semantics disqualifies all media data that is annotated, either continuously or discretely, with one of concepts, or not annotated at all; all other media data qualify. The operator variant for continuous media works on a finer level of granularity. It does not disqualify continuous media data entirely but rather only those temporal intervals that refer to one of the given concepts. The following definition introduces the operator formally for discrete and continuous query semantics.

Definition 8 — Query operator not
The query operator for discrete query semantics $not_d : 2^R \rightarrow 2^R$ is defined as follows:
$\forall a_m = (m, c, i) \in A : r_m = (m, \varepsilon) \in not_d(R_1)$ iff $\neg\exists\ r'_m = (m, i_1) \in R_1$.

The query operator $not_d : 2^R \rightarrow 2^R$ for continuous query semantics is defined as follows:
$\forall r_m = (m, i) \in R : (m, i) \in not_c(R_1)$ iff all of the ensuing conditions hold:

1. $\exists\ a_m \in A$.
2. if $i = \varepsilon$ then $\neg\exists\ r'_m = (m, i_1) \in R_1$.
3. if $i \neq \varepsilon$ then i is of maximal duration such that $\neg\exists\ r'_m = (m, i_1) \in R_1$ with $i_1 \neq \varepsilon$ and i_1 overlaps i.
4. if $i \neq \varepsilon$ then $\neg\exists\ r'_m = (m, \varepsilon) \in R_1$.

For instance, the video v depicted in Figure 2 does not qualify as a result for the query with discrete query semantics $not_d(retrieve_d(\text{“}\textit{thorac}\text{”}))$ as there exists an annotation relating a time interval to the concept “thorac”. However, the same query with continuous query semantics $not_c(retrieve_c(\text{“}\textit{thorac}\text{”}))$ returns the continuous references to the temporal interval of v ranging from the beginning of v to the start point of b and to the temporal interval of v ranging from the end point of b to the end of v among the results.

The operators we have introduced so far can be arbitrarily nested – as long as different query semantics are not mixed – and, therefore, provide a powerful means of uniform, content-based retrieval of media data. For instance, $and_c(and_c(retrieve_c(\text{“}\textit{thorac}\text{”}), retrieve_c(\text{“}\textit{medicament}\text{”})), not_c(retrieve_c(\text{“}\textit{complication}\text{”})))$ returns references to all media data of any type with contents dealing with the “thorac” and some “medicament” at the same time but with no incursion of any complications.

The MIB implements the query operators of both semantics introduced above as user-defined routines. These are based on the primitives for content-based retrieval offered by the COCOON DataBlade module. These primitives allow to retrieve all database rows that contain references a given concept of the concept hierarchy managed by COCOON. Moreover, a user-defined routine named `medToTempTable` has been provided which copies the results of a query to a temporary table. This table can then be simply joined with a table of the organization hierarchy of Section 3, thereby allowing for the mixed retrieval of media data according to technical characteristics and content.

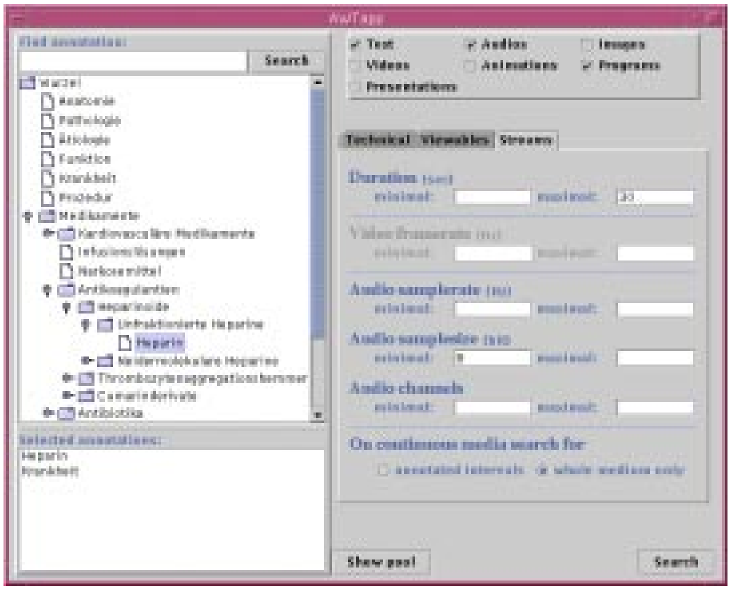
5 Application of the Media Integration DataBlade Module

The MIB has been successfully implemented and employed by our group for the management of the media data in the context of the Cardio-OP project. Recently, we have developed a graphical media browser which can be employed for the retrieval of media data managed by the MIB according to content and technical metadata in an intuitive manner. This browser exploits the features of the MIB with regard to the organization of technical metadata and the sophisticated content-based retrieval. Figure 3 shows a screenshot of the media browser. The left part of the browser shows the concept hierarchy (with German captions) managed by the MIB (see Section 4.1) out of which a user can select the desired set of concepts. The top right part of the browser allows to limit the search to specific media data types. According to this selection, the browser offers means to further narrow down the search by specifying constraints for those technical metadata applicable to the selected media data types. To the bottom right, the query semantics for the content-based retrieval as presented in Section 4.2 can be chosen. The query operators as presented in Section 4.3 implicitly are used by the browser when is comes to formulate the query to the DBMS. Finally, the search button activates the retrieval process and returns references to all qualifying media data using the uniform locator mechanism presented in Section 2. The returned locators can be used to access to actual media data transparently.

The media browser has been developed as a Java Bean to reuse in further applications that need media management and media browsing facilities. This component is currently reused for the development of a graphical annotation editor and an authoring tool for multimedia presentations. The graphical annotation editor aims at the comfortable creation and manipulation of annotations to discrete and continuous media data. It also uses the facilities provided by the MIB to insert and manage annotations. The media browser component is used to select media data for annotation and to browse the annotations. The authoring tool allows to build multimedia presentations using the ZYX document model [BK99] which has been developed in the context of the Cardio-OP project since standard models like SMIL, MHEG-5, and HyTime do not meet the project’s specific requirements for adaptation, reuse, and presentation neutrality of multimedia content [BKW99b]. The authoring tool utilizes the media browser component, and hereby the MIB, to browse and select media data for use in the multimedia composition. These tools show the application of the MIB for the integrated, uniform management and retrieval of media data of different types. Other applications exploiting the MIB for the intelligent management of media data can be imagined.

6 Conclusion and Outlook

Starting out from the need of multimedia information systems for the integrated and uniform management and access to media data of different media data types, we introduced the Media Integration DataBlade module as a generic component which unites media data type-specific database extensions under a common roof. We introduced the locator mechanism of the MIB allowing for the transparent, uniform access to media data stored in



☑ Text ☑ Audios ☐ Images
☐ Videos ☐ Animations ☑ Programs
☐ Presentations

Duration [sec] minimal: maximal: 20
Audio samplesize [bit] minimal: 8

Selected annotations: Heparin, Krankheit

On continuous media search for ○ annotated intervals ● whole medias only

Figure 3: Screenshot of the media browser component

various locations. We then explained how the media data is organized to support sophisticated search and retrieval according to technical characteristics effectively. We illustrated the facilities for content-based retrieval and showed that the MIB powerfully allows for the mixed retrieval of media data of various type by content and by technical metadata. We demonstrated the use of the MIB in applications requiring sophisticated, uniform management of media data of different type: browsing and content-based retrieval media data, management of annotations to media data, and multimedia authoring.

We plan to extend the MIB with support for further media data types. For this purpose, we evaluate database extensions for the management of audio data, MPEG streams, and animations. In addition to that, there is ongoing work on a streaming server placed on top of the MIB following the architecture of [BKL96] supporting the delivery of continuous media data over a network.

Acknowledgments We would like to thank Susanne Boll for her valuable comments and her great efforts to prepare and improve this version of the paper.

References

[AN97] D. A. Adjeroh and K. C. Nwosu. Multimedia Database Management – Requirements and Issues. *IEEE Multimedia*, 4(3), 1997.

[BG99] S. Bechhofer and C. Goble. Classification Based Navigation and Retrieval for Picture Archives. In *Proceedings 8th IFIP Conference on Data Semantics (DS-8)*, Rotorua, New Zealand, January 1999. Kluwer Academic Publishers.

[BK99] S. Boll and W. Klas. ZYX — A Semantic Model for Multimedia Documents and Presentations. In *Proceedings of the 8th IFIP Conference on Data Semantics (DS-8): "Semantic Issues in Multimedia Systems"*. Kluwer Academic Publishers, Rotorua, New Zealand, 5-8 January 1999.

[BKL96] S. Boll, W. Klas, and M. Löhr. Integrated Database Services for Multimedia Presentations. In S. M. Chung, editor, *Multimedia Information Storage and Management*. Kluwer Academic Publishers, Dordrecht, 1996.

[BKW99a] S. Boll, W. Klas, and U. Westermann. Exploiting OR-DBMS Technology to Implement the ZYX Data Model for Multimedia Documents and Presentations. In *Proceedings Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, Freiburg, Germany, March 1999. Springer.

[BKW99b] S. Boll, W. Klas, and U. Westermann. Multimedia Document Formats — Sealed Fate or Setting Out for New Shores? In *Proceedings IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, Florence, Italy, June 1999.

[dim98] dimedis. *The COCOON DataBlade Module Users Guide*. dimedis GmbH, Cologne, Germany, 1998.

[F$^+$95] M. Flickner et al. Query by Image and Video Content: The QBIC System. *IEEE Multimedia*, 28(9), 1995.

[JE98] H. Jiang and A. K. Elmagarmid. Spatial and Temporal Content-Based Access to Hypervideo Databases. *VLDB Journal*, 7(4), 1998.

[KA97] W. Klas and K. Aberer. Multimedia and its Impact on Database System Architectures. In P.M.G. Apers, H.M. Blanken, and M.A.W. Houtsma, editors, *Multimedia Databases in Perspective*. Springer, London, 1997.

[KGF99] W. Klas, C. Greiner, and R. Friedl. Cardio-OP — Gallery of Cardiac Surgery. In *Proceedings IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, Florence, Italy, June 1999.

[LR95] M. Löhr and T. C. Rakow. Audio Support for an Object-Oriented Database Management System. *Multimedia Systems Journal*, 3(5-6), 1995.

[NMB+98] C. Nastar, M. Mischke, N. Boujemaa, et al. Retrieving Images by Content: The Surfimage System. In *Proceedings 4th Int. Workshop on Multimedia Information Systems (MIS)*, Instanbul, Turkey, September 1998.

[OT93] E. Oomoto and K. Tanaka. OVID: Design and Implementation of a Video-Object Database System. *IEEE Transactions on Knowledge and Data Engineering*, 5(4), 1993.

[RNL95] T. C. Rakow, E. J. Neuhold, and M. Löhr. Multimedia Database Systems – The Notions and the Issues. In *Proceedings Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, Dresden, 1995. Springer.

[WS98] M. Wechsler and P. Schäuble. Metadata for Content-Based Retrieval of Speech. In A. Sheth and W. Klas, editors, *Multimedia Data Management*. McGraw-Hill, New York, 1998.