

Physical and Logical Integration of Data and Media Objects

Henrike Berthold

Technische Universität Dresden

Fakultät Informatik, Institut BDR, Professur für Datenbanken

01062 Dresden

E-mail: berthold@db.inf.tu-dresden.de

Abstract

Information systems in general manage formatted data, and most of them use databases to store them adequately. While these systems work well, there are new requirements now to improve them towards the inclusion of multimedia data, i.e. images, graphics, video, and audio. Usually, multimedia data are stored in specialized servers which can cope with the requirements of real-time storage and delivery. Applications being developed today, however, need the services of both databases and these media servers. This paper presents (1) an open architecture for the integration of media servers and databases into a single system and (2) the object-oriented database design for the logical integration of media and data objects.

1 Introduction and Motivation

Today, much work is dedicated to the handling, storing, and transporting of particular continuous-media data. But little work is invested to make the resulting systems interoperable, so that they can be integrated with existing systems or other components. In this paper, (1) an open architecture is presented for a Distributed Multimedia Database Management System (DMDBMS) that integrates databases – relational as well as object-oriented – with media servers or media-object stores and (2) the object-oriented database design is given for the logical integration of data and media objects.

The separate handling of formatted data and media data has several reasons. First, conventional databases often exist already, and they must be used in newly developed multimedia information systems as well, because the risk and the cost of building a completely new system are too high. Second, media data, especially the continuous types, must usually be stored on special hardware or on a separate server, because significant computing power is still required by the data transformations and even by the bandwidth used in delivery. Finally, the handling of media data is essentially different from that of formatted data.

The architecture consists of component systems and services. *Component systems* are conventional databases which store formatted data (FDB) like relational or object-oriented ones, and media servers or media-object stores. A *media server* can manage multimedia data, while a *media-object store* additionally offers database functionality, i.e.

persistence, data independence, etc. Media-object stores can also be called *multimedia databases*. *Services* are built on top of these component systems. These services form an integration middleware that is also called a *Distributed Multimedia Database Management System* (DMDBMS). They are characterized by their interfaces and functionalities. *Applications* can access the DMDBMS through the provided interfaces. *DMDBMS Clients* also use these interfaces. They provide a user interface to the DMDBMS.

The main characteristics of the DMDBMS are the following: (1) It is an open system and homogeneous, yet functionally different interfaces are used to describe services and component systems. (2) Media data and formatted data are handled in different ways but are logically integrated. (3) The component systems are tight coupled. (4) The object-oriented data model is used for the global schema.

The remainder of the paper is structured as follows. Section 2 presents the general architecture. In Sect. 3 the interfaces of the component systems are explained. The services that form the DMDBMS are introduced in Sect. 4. The resulting architectures are presented in Sect. 5. Section 6 deals with the logical integration of data and media objects in an object-oriented database. In Sect. 7 related work is viewed. The future work is short outlined in Sect. 8.

2 General Architecture

Interfaces can be found on different levels of the main architecture (Fig. 1):

- for DMDBMS applications,
- between DMDBMS and component systems and
- within the DMDBMS i.e. between the DMDBMS services.

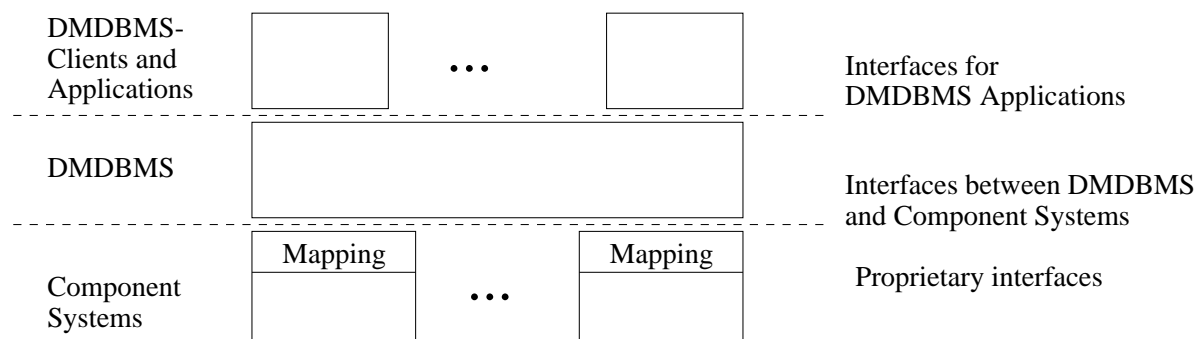


Figure 1: General architecture

Homogeneity of interfaces is achieved by defining a common syntax for all interfaces. The proprietary interfaces that have similar semantics must be mapped to this syntax. Since component systems and services are quite different, it is useful to have not just one, but several interfaces, which have different functionality. Each interface defines a set of functions, e.g. for boolean queries, fuzzy queries, multimedia access, and transaction management. This set can be organized in a specialization hierarchy to support tailoring and reuse. A component system is described by a set of control interfaces (and protocols) it maps to. A service uses interfaces which are called base interfaces to control or manage other services or component systems, and it offers control interfaces to be used by other services, the DMDBMS clients or applications.

Here, the set of interfaces is first divided into *Access Interfaces* (AIs) and *Interoperability Interfaces* (IIs). Interoperability interfaces are characterized by functions, and by a protocol that specifies allowed sequences of function calls. Access Interfaces are defined by functions only.

The advantages of using homogeneous interfaces are that (1) services can be composed as necessary, (2) services can be changed as long as the interfaces remain the same, (3) component systems can be added easily and (4) software reuse is possible.

3 Interfaces of Component Systems

Conventional database systems in general have an interface **AccessF** to access stored data, an interface **Session** to manage connections, an interface **TransTM** to control internal transactions, and an interface **TransRM** that allows control of transactions from the outside. A media server has at least an interface **AccessMM** to access multimedia data, an interface **Session** to manage connections, and often an interface **Search** for search over multimedia data. A multimedia database system must have **TransTM** and **TransRM** in addition.

Some interfaces, e.g. **Search**, have specializations, because there are many different search methods for multimedia data. On the contrary, the access to multimedia data is rather straightforward: its media-object identifier can be used. This is one reason for separating search from access for multimedia data. The other reason is that search and access are often performed in two consecutive steps: Access to multimedia data requires a protocol and a tight connection, while search does not. Other interfaces, e.g. the transaction interfaces, have a fixed set of functions and no specialization. They stick to the DTP standard as defined by X/Open. The transaction interfaces, **AccessMM**, and **Session** are interoperability interfaces, while the other ones are access interfaces. In order to mark this difference, the interface names are qualified by II (for interoperability interfaces) or AI, respectively. In Table 1 all component-system interfaces are summarized. Some of them are described in the following subsections.

II-Session	connection management
AI-Search	search over multimedia objects
II-AccessMM	access to multimedia objects
AI-AccessF	access to formatted data
II-TransTM	control transactions
II-TransRM	control transactions of resources (e.g. a database system) from the outside

Table 1: Functional interfaces

3.1 II-Session

The two functions needed here are open and close. They help to authorize a user or application and to attach function calls to a session/connection. The protocol has two states: Init(0) and Opened(1). An open call causes the state transition: $0 \rightarrow 1$, the close call reverses it.

3.2 AI-AccessF

The **AI-AccessF** interface can be defined in different ways. First, a generic interface based on standardized languages can be used. These languages are SQL (SQL-2 or SQL-3) for relational database systems and ODL/OQL defined by the ODMG [5](ODMG 1.2 or ODMG 2.0) for object-oriented ones.¹

- **AI-AccessSQL**
exec_sql(sql-statement) and
- **AI-AccessODMG**
exec_odmg(odmg-statement).

Using a generic interface is a general and universal approach. A single interface can be used for all databases with the same data model. It is independent of schema modifications as well as change of user requirements. But there remains the mismatch between relational and object-oriented technology which leads to at least two of these interfaces.

The other approach is to think of **AI-AccessF** as a description of the schema. For an object-oriented database system, the description consists of all methods that offer access to the data. In a relational database system, such a description is provided by an application program that implements access functions. This variant requires a fixed schema and thus bears less flexibility. If users have new requirements, it is necessary to change the mapping, for instance by implementing new methods or functions.

3.3 AI-Search

There are many search methods for multimedia data; a general overview can be found in [15]. Content-based information retrieval or media information retrieval are usually tailored to a certain media type, whereas the attribute-value search based on meta-data is applicable to all media types. Media information retrieval is fuzzy and therefore ranks the results. This yields a list, while attribute-value search leads to boolean retrieval and thus produces a set.

In Table 2, some search methods are described by the types of their input and output values. This is exactly the information needed for the integration into a DMDDBMS.

3.4 II-AccessMM

Media server store single media objects. Media objects are addressed by their media-object identifiers. Each media object has a determined media type, i.e. image, graphic, audio, or video. The hierarchy of single media objects is shown in Fig. 2.

The interface AccessMM contains of two parts: management functions which are independent of media objects e.g. to get informations about the media server and methods on media objects. The latter follow a protocol. So some methods on media objects can only be called in a certain state. The states reflect different access modes. All media objects can be downloaded, but only continuous media objects can be streamed. So the states *Unspecified*, *Opened* and *Data Defined* are appropriate for all media objects, whereas the states *Stream Defined* and *Streaming* are only appropriate for continuous media objects.

¹The latter, however, is not widely used yet. Up to this date, object-oriented database systems offer many different query and data definition languages and do not even have a common object model.

Input (Search argument)	Output (Search result)	Comment
type_x_media_object	list(type_x_media_object)	sample object or pattern
type_y_media_object	list(type_y_media_object) with $x \neq y$	
language expression	set(media_object)	media objects must be described by a language expression (for instance through prolog rules)
formatted data	set(media_object)	media objects must have a schema, e.g. attached attributes
keywords	list(media_object)	media objects must be described with keywords

Table 2: Input and output values of search engines

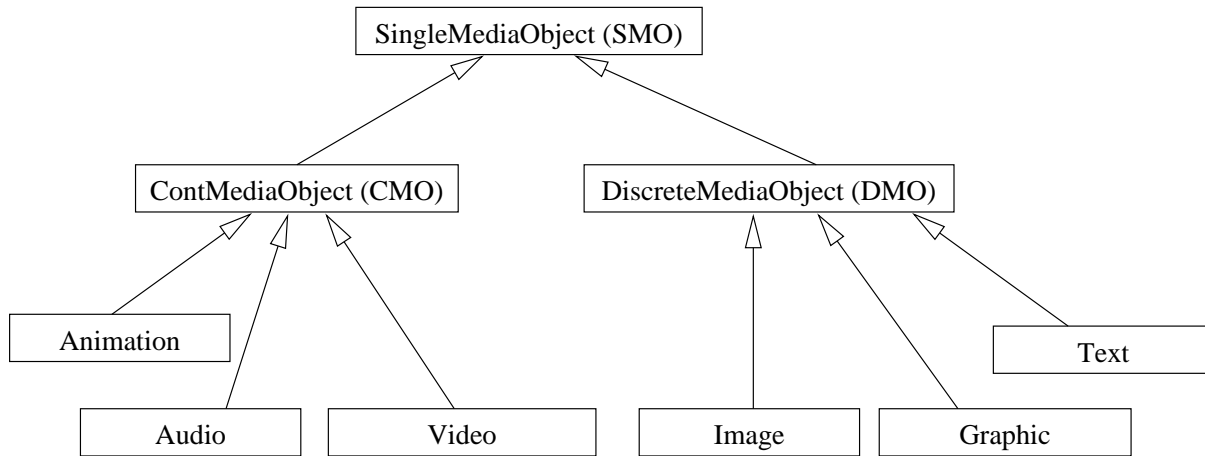


Figure 2: Hierarchy of single media objects

A media type dependent set of parameters is used to specify the desired quality of a media object. For example the set of parameters for the media type image consists of image format e.g. gif, part, resolution, color depth, color and dpi. Additionally parameters are necessary that specify the quality of a stream i.e. service type (guaranteed, best effort or statistical), the degree of interactivity, delay and error-rate.

4 Services

In this section several services are introduced. An overview of all services, together with their control interfaces, base interfaces, and base services, is given in Table 3. A *base service* is a service which is used by the service under consideration.

The service that allows to execute distributed transactions is called the *Transaction Coordinator*. It requires all component systems to have an interface **II-TransRM** supporting the two-phase commit protocol.

If a user wishes to integrate two or more conventional database systems, a additional database management system is used to manage the global schema. This service is called *Formatted-data Integration Service*.

If more than one media component system is used, a *Media Manager* has to be em-

ployed. A single media component system can be handled through its own client software, but with more than one, a service is needed that negotiates quality of service and controls media transport.

The *Media Naming Service* is responsible for a global naming of media objects, the *Media Integration Service* stores and manages relations between media and formatted data, and the *Media Search Services* use information-retrieval methods to find media objects.

There are two new interfaces related to these services, namely **AI-NameMM** and **AI-IntegratedAccess**. **AI-NameMM** is the interface of the Media Naming Service, **AI-IntegratedAccess** is that of the Media Integration Service.

Formatted-data Integration Service Control interface Base interface Base services	Management of the global schema for formatted data AI-AccessF AI-AccessF none
Media Integration Service Control interface Base interface Base services	Management of the global schema that includes relations between media and data objects AI-IntegratedAccess II-AccessMM, AI-AccessF Formatted-data Integration Service, Media Manager, Media Search Service
Media Manager Control interface Base interface Base services	Access to media objects; Negotiation, reservation, and observation of quality; Transport of media objects according to that quality II-AccessMM II-AccessMM Media Naming Service
Media Naming Service Control interface Base interface Base services	Global naming of media objects AI-NameMM II-AccessMM none
Media Search Service Control interface Base interface Base services	Information retrieval in media objects AI-Search II-AccessMM Media Naming Service
Transaction Coordinator Control interface Base interface Base services	Control and management of (distributed) transactions II-TransTM II-TransRM none

Table 3: Services

4.1 Media Search Service

It is not possible to apply one search method to all media servers because (1) search methods supported by media servers are quite different and (2) there is no standardized

query language and even no standardized meta-data model for media data.

To make a search possible that produces a list or set of relevant media data stored in different media servers, it is necessary to use an external service. Since meta-data stored in media servers are quite different, an Information Retrieval service called Media Search Service which indexes the media data itself seems most appropriate.

The interface **AI-SearchIR** of the Media Search Service has operations *search* and *order*. *Search* determines a list of media objects similar to a given sample media object. The number of media objects in the result list can be limited. The media objects are ordered by degree of similarity within the result list. *Order* sorts a given set of media objects by degree of similarity to the sample media object.

5 Architecture

A full architecture can be build for a system that includes conventional databases as well as multimedia databases (see Fig. 3). If only one conventional database system is used, the Media Integration Service accesses this component system directly, and the Formatted-data Integration Service can be omitted. If only one multimedia database is used, the Media Manager and the Media Naming Service should be omitted, and the client of the multimedia database system should be used instead. If only media servers and just one conventional database system are used, there is no need for a Transaction Coordinator.

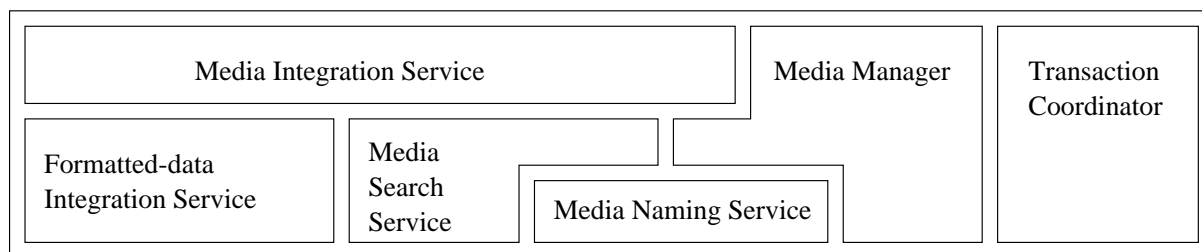


Figure 3: Full DMDBMS

6 Media Integration Service

The Media Integration Service implements a global object-oriented schema that represents media and data objects as well as relations among media objects and between media and data objects. The interface AI-IntegratedAccess of the Media Integration Service has functions to query and manipulate the data that are represented by the global schema. The object-oriented paradigm is used because some of its concepts are very useful for modeling media data e.g. encapsulation, complex types and inheritance. The instances of the relations i.e. the object identifiers of related objects are stored in an internal database.

The design presented here is based on the CROQUE object model [21] that gives a formal data model for the ODMG standard [6] with the following changes.

- ODMG's structured objects are atomic objects in CROQUE. Structures are structured values i.e. literals.
- Types and Classes are distinguished.

- An object can be an instance of multiple types (at the same time and at various points in its lifetime).
- Object types are arranged in a lattice, such that object-preserving projections need not be restricted to named supertypes of the database schema. The type system builds a lattice of named and unnamed object types below the one basic sort of atomic objects which is the domain of OIDs.

6.1 Object-Oriented Database Design

SingleMediaObjects in the database model are only proxies for real single media objects. When a user has found a set of SingleMediaObjects using the search facilities i.e. the database query language, he/she uses the method `ident` of SingleMediaObject (see Fig. 4) to get the MOID. With this MOID he/she can access the real single media object via the media manager.

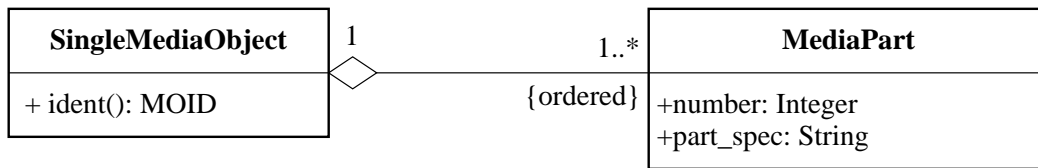


Figure 4: SingleMediaObject and MediaPart in UML

A single media object consists of several parts e.g. a video is composed of frames. These parts can be addressed separately e.g. a video frame can be specified by its number. The parts of media objects and their specifications in time and space are described in a media data model. Since there exists no standard media data model, we simply define MediaPart objects (see Fig. 4) which are associated to SingleMediaObjects and hold a value that specify the part.

Single media objects can be modeled in different ways in the global schema (see Sect. 6.1.1). The data objects which are defined in the schema of the formatted-data integration service are simply copied to the global schema. Relations between media and data objects and within media objects must be modeled in the global schema, too. In this section possible relations are shortly characterized.

Multimedia relation (UML description see Fig. 5): A multimedia relation relates parts of single-media objects in space and/or time under synchronization constraints. It is a many-to-many relation. The parenthesis express that this relation need not be modeled in the schema.

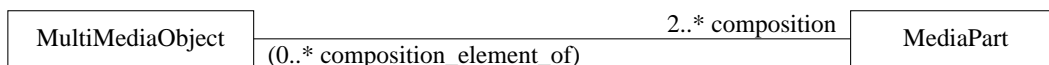


Figure 5: Multimedia relation

Presentation relation (UML description see Fig. 6): A presentation relation relates parts of single-media objects in space and/or time under synchronization constraints. It specifies additionally interaction points. An interaction point allows user-actions. It is a many-to-many relation.

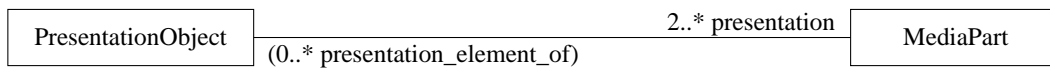


Figure 6: Presentation relation

Alternative relation (UML description see Fig. 7): An alternative relation relates media objects of 'similar semantics' e.g. a video object and a text object with the textual description of the video's content. It is a many-to-many relation.



Figure 7: Alternative relation

Hypermedia relation (UML description see Fig. 8): A hypermedia relation relates media object parts with a common topic. It is a many-to-many relation.

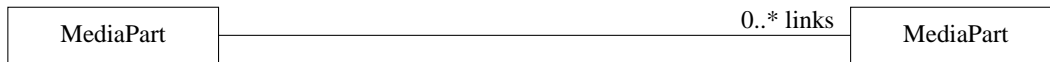


Figure 8: Hypermedia relation

Representation relation (UML description see Fig. 9): Media objects represent persons, events and objects of the real world, whose data are stored in data objects.

Description relation (UML description see Fig. 10): Data objects can store informations that describe media objects e.g. producer or date of production. This is a many-to-many relation.

Similarity relation: The similarity relation relates every two media objects. This relation has an associated number value that represents the degree of similarity.

Multimedia and presentation relations are compositions of parts of single media objects. Everyone can compose his/her presentation or multimedia objects using authoring tools. These objects can be stored in the media integration service, but it is also possible to store them somewhere else. We assume that they are not stored in the media integration service. Description and representation relations are not separated here. We call all relations between media and data objects representation relations. So in the global schema design the representation, hypermedia, alternative and sample relation are considered.

6.1.1 Modeling of Single Media Objects

The hierarchy of single media objects has been shown in Fig. 2. There are some possibilities to model this hierarchy with an object-oriented database schema:

type1 Definition of one basic single media object type and no subtypes,

type2 Definition of a type hierarchy for media objects,

class1 Definition of a class hierarchy with classes for every media type e.g. class SingleMediaObject and class Video

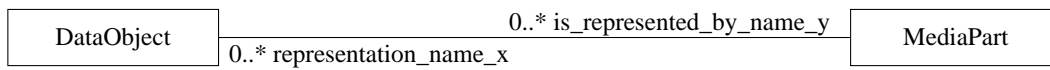


Figure 9: Representation relation

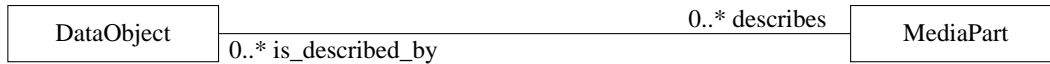


Figure 10: Description relation

class2 Definition of a basic class SingleMediaObject

class3 Definition of no class

All the six combinations of class and type hierarchies are possible. The definition of a class hierarchy (class1) has the advantage that the media objects are grouped in classes according their real types which is directly supported by the management part of the component interface II-AccessMM. These classes can be used in queries. If there is only one basic class defined (class2), only this basic class can be used in queries. If there is no class at all (class3) then the general set is not queriable. The definition of one type (type1) is used because all single media objects are only placeholders for real media objects and so the type specifications in the database schema are equal. The definition of a type hierarchy with equal types (type2) is not useful. So the combination (type1, class1) is used.

Modeling in ODMG/CROQUE

There are two ways to integrate single media object types in the ODMG type hierarchy - (a) as built-in type below Object_type or (b) below Atomic_object. The advantage of the first approach is, that the type is uniform in all object-oriented database management systems, but to achieve this it must be included in the standard. The definition would look as follows

```

interface SingleMediaObjectFactory: ObjectFactory {SingleMediaObject new(MOID); ...};
interface SingleMediaObject: Object {MOID ident()};
  
```

The second approach is realizable without standardization efforts and hence it is used here. The schema in ODMG has the interface TSingleMediaObject and a class for every single media object type i.e. SingleMediaObject, ContMediaObject, Animation, etc.

```

interface TSingleMediaObject {MOID ident()};
class SingleMediaObject: TSingleMediaObject (extent SingleMediaObjects) {};
class ContMediaObject extends SingleMediaObject: TSingleMediaObject (extent
ContMediaObjects) {};
class Animation extends ContMediaObject: TSingleMediaObject (extent Animations) {};
...
  
```

The CROQUE schema for the second approach looks as follows. It expresses the same than the ODMG schema. The extents of the classes are not specified here, because they are managed automatically.

```

type TSingleMediaObject = interface{ident: MOID};
  
```

```

class SingleMediaObject: TSingleMediaObject;
class ContMediaObject: TSingleMediaObject some SingleMediaObject;
class Animation: TSingleMediaObject some ContMediaObject;
...

```

6.1.2 Modeling of the representation relation

The representation relation relates data and media objects. Relationships are used to express these relations in the object model. There are many possibilities, some of which are discussed below.

(rep1) General relationships that are inherited by all data objects and single media objects

The type TDataObject would get the relationship *media: set<SingleMediaObject>*, and the type TSingleMediaObject would get the relationship *data: set<DataObject>*.

General relationships are universal but have no semantics. All data objects must inherit this relation from the general type DataObject.

(rep2) Named general relationships

Named general relationships have only few semantics and most of the relationships have empty object sets.

(rep3) Specific relationships between data objects of a certain class and media objects of a certain class

The relationship is added to the data object type that is associated with the class. If a special subclass is necessary, this subclass must be added in the appropriate database. For media objects a subtype with the relationship and a class with this type must be created.

The cardinality, characteristics i.e. shared/private, dep/indep, close/open² and inverse relationships can be expressed directly. So specific relationships have high semantics.

Since the semantics can be expressed best with (rep3), this variant is used.

Modeling in ODMG/CROQUE

In the ODMG object model an object can only be in one class. So if there are two subclasses e.g. Passport_photo and Photo of class Image, then an object which is Passport_photo cannot be in Photo. So there is no media object sharing.

Example 1: Every person should get an associated passport photo.

This example can be realized in ODMG and CROQUE easily. Only the type specifications of relationships are different. In ODMG class names and in CROQUE type names must be used.

ODMG:

```

interface TPassport_photo: TSingleMediaObject {relationship Person photo_of inverse
Person::photo;};
class Passport_photo extends Image: TPassport_photo {};

```

²These characteristics do not exist in ODMG 2.0 and CROQUE.

```
class Person ...{... ; relationship Passport_photo photo inverse Passport_photo::photo_of;}
```

CROQUE:

```
type TPassport_photo = interface: TSingleMediaObject{photo_of: TPerson inverse photo};
class Passport_photo: TPassport_photo some Image;
type TPerson = interface ...{photo: TPassport_photo inverse photo_of};
class Person: TPerson ...;
```

Example 2: Every person should be related to all media objects in which he/she is to be seen i.e. video and image.

It is necessary to define two subclasses, one below class Image and one below class Video. Because of the strict separation of types and classes in CROQUE, it is only necessary to define one type TVisualRep in the CROQUE schema and use this type to specify the relationship. In ODMG two interfaces for the two classes must be defined, because class names are used to specify relationships.

ODMG:

```
interface TImageRep: TSingleMediaObject {relationship set<Person> in
inverse Person::seen_in_images;};
interface TVideoRep: TSingleMediaObject {relationship set<Person> in
inverse Person::seen_in_videos;};
class ImageRep extends Image: TImageRep {};
class VideoRep extends Video: TVideoRep {};
class Person ...{... ; relationship set<ImageRep> seen_in_image inverse ImageRep::in;
relationship set<VideoRep> seen_in_video inverse VideoRep::in; }
```

CROQUE:

```
type TVisualRep = interface: TSingleMediaObject{seen: set of TPerson inverse seen_in};
class ImageRep: TVisualRep some Image;
class VideoRep: TVisualRep some Video;
type TPerson = interface ...{seen_in: set of TVisualRep inverse seen};
class Person: TPerson ...;
```

6.1.3 Modeling of the sample relation

The relation sample expresses the similarity between every two single media objects. The similarity is represented by a number - typically a real number between 0 and 1. 1 means that the two single media objects are identical and 0 means that there is no similarity at all. This relation may not be expressed statically. It is calculated using an index (see [22]).

Generally there are two possibilities to model this relation: object methods or generic operations. Since generic operations or operators must be applicable to all types, object methods were chosen. The following two different approaches for object methods are possible.

(sam1) Method *resembles* with one parameter of type TSingleMediaObject and a return value of type Rank that extends type TSingleMediaObject

(1) **Query example using method *resembles*:**

```
select p.name
```

from Persons p

where p.passport_photo.resembles(photo_of_searched_person) > 0.5;

(2) Query example - get list of Rank values:

```
select p.name, p.passport_photo.resembles(photo_of_searched_person) as rank
```

```
from Persons p
```

```
order by rank;
```

(3) Query example - get list of single media objects:

```
select p.name
```

```
from Persons p
```

```
order by p.passport_photo.resembles(photo_of_searched_person);
```

(4) Query example - use of an example image stored in the database:

```
select p
```

```
from Person p, Person q
```

```
where q.name='Meier'
```

```
order by p.passport_photo.resembles(q.passport_photo);
```

Disadvantages:

(1) The direct handling of rank values is necessary.

(2) Existing information i.e. the order of objects according the degree of similarity can get lost in queries e.g. in query 1.

(sam2) Method *similarity* that returns a list of single media objects

It can be **(sam2-a)** a method of collections of single media objects i.e. a class method of `SingleMediaObject`³ or **(sam2-b)** a method of the example single media object that extends type `TSingleMediaObject`.

(1) Query example for a:

```
select plist.photo_of.name
```

```
from passport_photo.similarity(photo_of_searched_person) as plist
```

```
where plist.photo_of.date_of_birth > 01.01.1980
```

(2) Query example for b:

```
select simlist.photo_of.name
```

```
from photo_of_searched_person.similarity(Persons.passport_photo) as simlist
```

```
where simlist.photo_of.date_of_birth > 01.01.1980
```

Since most object data models do not know class methods or characteristics of collections, sam2-a is not used here. Sam1 expresses the same as sam2-b, but sam2-b assumes that the example is part of the database, so sam-1 is used.

Modeling in ODMG/CROQUE

The method *resembles* is added to type `TSingleMediaObject` to express this relation. It can be defined easily in ODMG and CROQUE.

ODMG:

```
interface TSingleMediaObject { ...; Rank resembles();};
```

CROQUE:

```
type TSingleMediaObject = interface { ..., resembles: Rank};
```

³ODMG 2.0 and CROQUE do not allow the specification of class characteristics.

6.1.4 Modeling of the hypermedia relation

A reference model for hypermedia relations is the Dexter Hypertext Reference Model [13]. It introduces three layers for a hypertext system (see Fig. 11), the run-time layer, the storage layer and the within-component layer. It is mainly focussed on the storage layer which describes a database containing components and links. Components are seen as containers without any structure. The within-component layer has the knowledge about the internal structure of components. The anchoring mechanism is used to address locations within components. Presentation specifications are used to encode how components are to be presented.

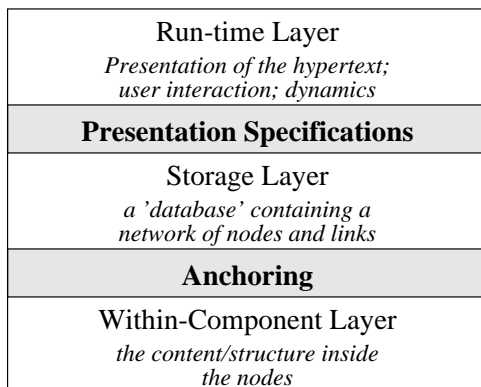


Figure 11: Dexter Layers

In [13] the storage layer model and the run-time layer model are described. Here the storage layer model is of interest.

A hypertext is described as a finite set of components together with a resolver and an accessor function. The basic entity is the component, which is either an atom, a link or a composite entity. An atom is a primitive component whereas a composite entity is composed of multiple components. A link represents a relation between components. It is a sequence of two or more endpoint specifications. Every component has a globally unique identifier (UID). The accessor function is responsible for accessing a component using its UID. The resolver function is used for resolving a UID from a component specification. An anchor consists of an anchor id and an anchor value, which is not interpreted within the storage layer. Link endpoints are specified with so-called specifiers, which consist of a component specification, an anchor id, a direction (i.e. FROM, TO, BIDIRECT, NONE) and a presentation specification. Components have associated component information which are a sequence of anchors that index into the component, a presentation specification and a set of attribute/value pairs.

In the Dexter Reference Model the base structure is a component, which can be an atom, a link or a composite composed of components. Here a single media object is an atom. Compositions of single media objects i.e. composites of atoms are not considered because multi-media and presentation relations were introduced beside hypermedia relations. Since in the Dexter Model components are related to each other, there is also a hierarchy of links possible. But a link hierarchy can always be represented in a flat form (see Fig. 12). A component has in the Dexter Model associated information i.e. attributes, a presentation specification and anchors. The presentation specification is not necessary, because only the existence of a link between single media objects is modeled,

not the fact how the links are presented. Media part objects, that describe parts of single media objects, are used instead of anchors. One link attribute that holds the link name is used. Due to the additional relation attribute, links must be objects. MediaLink is the type of these objects. Fig. 13 shows the modified UML description for hypermedia relations.

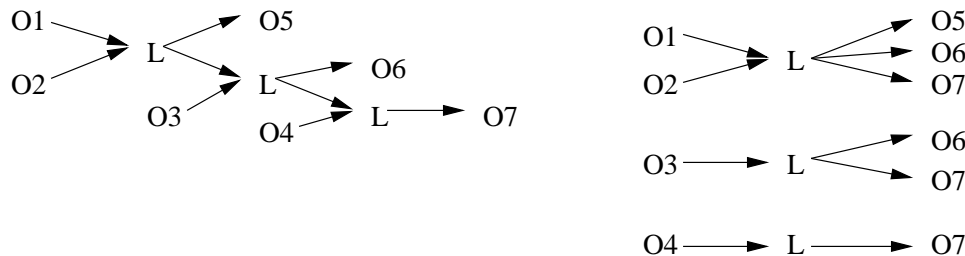


Figure 12: Link hierarchy



Figure 13: Hypermedia relation (modified)

All link directions can be expressed with this model. An single media object with an bidirect link is contained in link_from and link_to object-set of the link object.

Modeling in ODMG/CROQUE

This relation is represented by a type and a class MediaLink. The only difference between ODMG and CROQUE is the type specification in relationships (class vs. type). So only the ODMG schema is presented.

```
interface TMediaLink {attribute string name; attribute set<MediaPart> link_to;
relationship set<MediaPart> link_from inverse MediaPart::link_to; };
interface TMediaPart {...; relationship set<MediaLink> link_to inverse MediaLink::link_from;}
class MediaLink: TMediaLink {};
class MediaPart: TMediaPart {};
```

6.1.5 Modeling of the alternative relation

An alternative aims to offer a 'similar semantics' in a lower quality. A quality order between alternatives does not exist, but it is possible to use the object size to judge about the quality. So every media part object can reference a lower_quality and a higher_quality media part object.

Modeling in ODMG/CROQUE

This relation is represented by relationships between media parts. Again the difference between ODMG and CROQUE is the type specification in relationships. So only the ODMG schema is presented.

```
interface TMediaPart {...; attribute MediaPart lower_qual inverse MediaPart::higher_qual;
attribute MediaPart higher_qual inverse MediaPart::lower_qual;}
class MediaPart: TMediaPart {};
```

7 Related Work

Homogeneous interfaces are based on standards. One such standard for databases is the Remote Database Access (RDA) standard, which has been specified by the International Standards Organization (ISO). It has the status of an international standard since 1993. RDA is a service in layer 7 of the ISO/OSI reference architecture. It is an asymmetrical protocol like the RPC protocol, and it consists of a generic part plus a language-dependent special part for the execution of SQL commands. It is divided into functional units, which are comparable to functional interfaces, i.e. II-Session, II-TransTM, and AI-AccessF (generic using SQL). This standard can be used for session management and access to formatted data.

A lot of work has been done concerning single services and multimedia databases/media servers. The integration of some conventional databases – needed to build the formatted-data integration service – is well examined. An integration based on CORBA is shown in [8, 9]. An overview of Quality-of-Service architectures, which can be used for the Media Manager, is given in [2]. At the moment there are many research activities in this area. Transaction Coordinators based on X/Open's DTP standard are already in use, and many conventional database management systems support the XA interface. The Object Transaction Service specified by the OMG as a CORBA service is compatible with this standard, too. Information retrieval of multimedia data is also a current research topic. [3] presents a model and an architecture for distributed information retrieval, which can be used to build a Media Search Service. A lot of research is done in the area of multimedia databases and media servers [16, 20, 1, 12, 18, 19, 11, 23].

So far, only a few proposals for systems which include conventional databases and media server can be found.

HERMES [24] and TSIMMIS [7] are based on mediators to integrate different data sources e.g. classical databases and pictorial data. The aim of these projects is the easy integration of data sources in a dynamic system so that all data can be queried together. The data can be read but not modified in such a system. In TSIMMIS and HERMES all data are handled in the same way, so no access methods dedicated to either media data or formatted data are provided. In TSIMMIS media data sources are classified and a static set of meta-data are extracted from media data and exported. The classification and extraction process can be different for all media data sources. So querying does not affect all data sources in the same way. In HERMES software packages e.g. for face recognition can be integrated too, but these services are not coupled to data sources, so calculations must be done at run-time, which can result in long query execution times. An indexing hierarchy cannot be built in advance.

An approach to store multimedia and traditional data in one system is made in Garlic, developed by the IBM Almaden Research Center [4]. It includes heterogeneous data sources such as databases, files, text managers, and image managers. In this project a unified schema based on an extension of the ODMG-93 data model and an object-oriented dialect of SQL are employed. Within Garlic, only internal protocols are used, so it is not an open system. In Garlic special requirements of media data except querying are not addressed. There is no access interface and issues of media data transport are not discussed.

Another approach is the Multiware Database [25]. The federated object-oriented database management system contains objects which describe multimedia documents. An information object contains data that describe content, structure, and synchronization as-

pects, whereas a presentation object describes features for the presentation. Information objects and presentation objects are related in a one-to-many relationship. Multimedia objects are stored on special servers. Multiware is based on CORBA. All meta-data are stored in a federated (distributed) object-oriented database system. Media servers are used by Multiware, but they are not part of it. The meta-data of a media object are not handled by itself. The presentation features are negotiated via a QoS protocol.

[17] presents an early approach to multimedia databases. An object-oriented data model is introduced that integrates the different conceptual models of the media. Each medium has its own conceptual model and its own database. An object base defines the relations between these media databases. Three Multimedia Database Management Architectures are introduced. A single DBMS architecture manages different media data together with the relations. A primary-secondary architecture consists of two kinds of DBMS. A secondary DBMS manages data of one media type. All secondary DBMSs and the object base are controlled by one primary DBMS. The federated architecture consists of open-member DBMS. These open DBMS can communicate through their external interfaces in order to process queries and updates. The object base can be centralized or decentralized. The primary-secondary architecture comes closest to the architecture discussed above in this paper. Component database systems can be seen as secondary DBMS and the DMDBMS is the primary DBMS. But the description given in [17] is not detailed enough to judge completely.

Commercial objectrelational databases e.g. Informix Universal Server, Oracle 8 or IBM Universal Server enrich the relational data-model with object-oriented concepts. Specifically they allow users the definition of complex data-types and functions on objects of these data-types. But the storage is fixed. It is not possible to add a special storage module for media data. So media objects can be added based on a definition of an appropriate complex data-type, but they are not handled and stored in a specific way according their special requirements e.g. real-time transport.

The integration of information retrieval in databases is investigated in [10] and [14]. [14] presents an OQL-oriented query language called P-OQL for PCTE that is based on the object model of PCTE. This object model consists of objects and links between them. Term based text retrieval facilities are included directly in the query language with three additional operators. These operators are D_vector that calculates the document description, Q_vector that calculates the query description and sim that calculates the similarity.

A probabilistic relational algebra is presented in [10]. A weight which expresses the probability is added to each tuple and the handling of this weight in operations of the relational algebra (projection, selection) is shown. A new operation that is called \wedge -join for probabilistic ranking is introduced.

8 Status and Future Work

This paper describes work in progress. The specification of the Media Integration Service will be completed. A compiler for OQL which decomposes queries into three parts (for the schema of the Formatted-data Integration Service, the internal schema of the Media Integration Service, and the IR server) and re-integrates the answers is being developed.

A prototype of a DMDBMS and a DMDBMS client will be implemented based on two component systems: a relational database and a video server.

References

- [1] Donald A. Adjeroh and Kingsley C. Nwosu. Multimedia Database Management – Requirements and Issues. *IEEE Multimedia*, 4(3):24–33, 1997.
- [2] Cristina Aurrecochea, Andrew Campbell, and Linda Hauw. A Survey of Quality of Service Architectures. Technical report, University of Lancaster, 1995.
- [3] Christoph Baumgarten and Klaus Meyer-Wegener. Towards a Scalable Networked Retrieval System for Searching Multimedia Databases. In *Proc. ACM SIGIR Workshop on Networked Information Retrieval*, 1997.
- [4] M.J. Carey et al. Towards Heterogenous Multimedia Information Systems: The Garlic Approach. www, 1995. <http://www.almaden.ibm.com/cs/garlic/ride-dom95.html>.
- [5] R.G.G. Cattell. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, 1994.
- [6] R.G.G. Cattell et al., editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.
- [7] Sudarshan Chawathe et al. The tsimmis project: Integration of heterogeneous information sources. In *Proc. of IPSJ Conference*, 1994.
- [8] Asuman Dogac, Cevdet Dengi, and M. Tamer Özsu. Building interoperable databases on distributed object management platforms, 1996. <http://web.cs.ualberta.ca/ozsu/publications.html>.
- [9] Asuman Dogac et al. METU Interoperable Database System. www, 1995. <http://www.srdc.metu.edu.tr/mind/publications.html>.
- [10] Norbert Fuhr. A Probabilistic Relational Model for the Integration of IR and Databases. In *Proceedings of ACM SIGIR*, pages 309–317, 1993.
- [11] D.James Gemmell et al. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, pages 40–49, May 1995.
- [12] Sreenivas Gollapudi and Aidong Zhang. Netmedia: A Client-Server Distributed Multimedia Database Environment. www, 1996. <http://www.cs.buffalo.edu/pub/tech-reports/README.html>.
- [13] F. Halasz and M. Schwartz. The dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, Feb. 1994.
- [14] Andreas Henrich. Document retrieval facilities for repository-based system development environments. In *Proceedings of ACM SIGIR*, pages 101–109, 1996.
- [15] John Z. Li, M. Tamer Özsu, and Duane Szafron. Query languages in multimedia database systems. Technical report, Department of Computing Science, The University of Alberta, Canada, 1995.
- [16] Ulrich Marder and Günter Robbert. The kangaroo project. In *Proceedings of the Third Int. Workshop on Multimedia Information Systems*, 1997.

- [17] Yoshifumi Masunaga. Multimedia Databases: A formal framework. In *Proceedings of the IEEE Computer Society Symposium on Office Automation*, pages 36–45, 1987.
- [18] A. Desai Narasimhalu. Multimedia Databases. *Multimedia Systems*, 4(5):226–249, 1996.
- [19] M. Tamer Özsu, Duane Szafron, Ghada El-Medani, and Chiradeep Vittal. An onject-oriented multimedia database system for a news-on-demand application. *Multimedia Systems*, 3(5):182–203, 1995.
- [20] Thomas C. Rakow, Wolfgang Klas, and Erich J. Neuhold. Research on Multimedia Database Systems at GMD-IPSI. *IEEE Multimedia Newsletter*, 4(1), 1996.
- [21] Holger Riedel and Marc H. Scholl. The croque-model: Formalization of the data model and query language. Technical report, Universtität Konstanz, 1996.
- [22] Peter Schäuble. *Multimedia Information Retrieval*. Kluwer Academic Publishers, 1997.
- [23] Prashant J. Shenoy, Pawan Goyal, Sriram S. Rao, and Harrick M. Vin. Symphony: An Integrated Multimedia File System. Technical report, Department of Computer Sciences, University of Texas at Austin, 1997.
- [24] V.S. Subrahmanian et al. HERMES: A Heterogeneous Reasoning and Mediator System. www, 1994. <http://www.cs.umd.edu/projects/hermes/overview/paper/index.html>.
- [25] Carlos M. Tobar and Ivan L.M. Ricarte. Multiware Database: A Distributed Object Database System for Multimedia Support. www, 1995. <http://www.dca.fee.unicamp.br/ricarte/Papers/papers.html#iciims96>.