

A flexible architecture for the integration of media servers and databases

Henrike Süß

Database Group, Department of Computer Science,
Dresden University of Technology, Germany,
suess@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 a flexible architecture for the integration of media servers and databases into a single system.

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, an 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.

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 (1) the use of homogeneous, yet functionally different interfaces to describe services and component systems, (2) the special handling of media data which concerns search and media data access and (3) the tight coupling of the component systems, so that read and write operations as well as transactions are supported.

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.

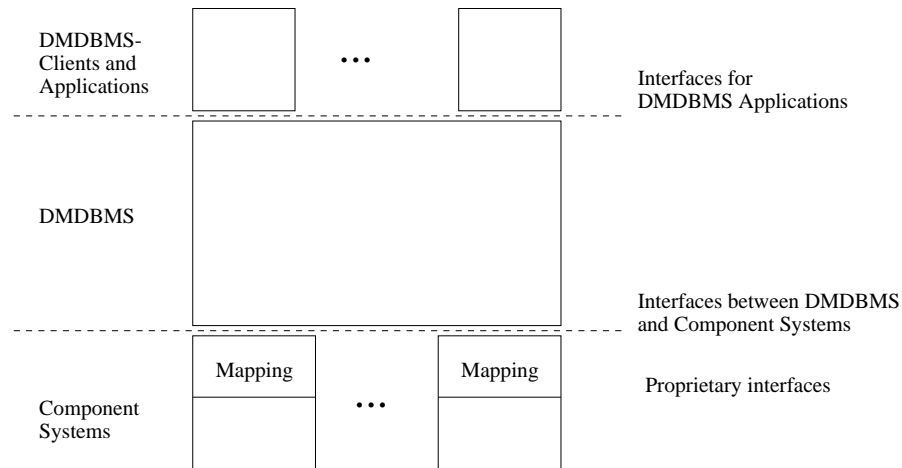


Fig. 1. General architecture

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.

II-Session	connection management
AI-Search	search over multimedia objects
II-AccessMM	access to multimedia objects
AI-AccessF	access to formatted data
AI-AccessDev	access for multimedia devices (input devices: camera, ... ; output devices: screen, ...)
II-TransTM	control transactions
II-TransRM	control transactions of resources (e.g. a database system) from the outside

Table 1. Functional interfaces

In the following subsections some interfaces are described.

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 [11]. 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 DMDBMS.

¹ The 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
text_object	list(text_object)	fulltext matching or linguistic similarity media objects must be described textually
text_object	list(media_object)	
type_x_media_object	list(type_x_media_object)	sample object or pattern
type_x_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

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.

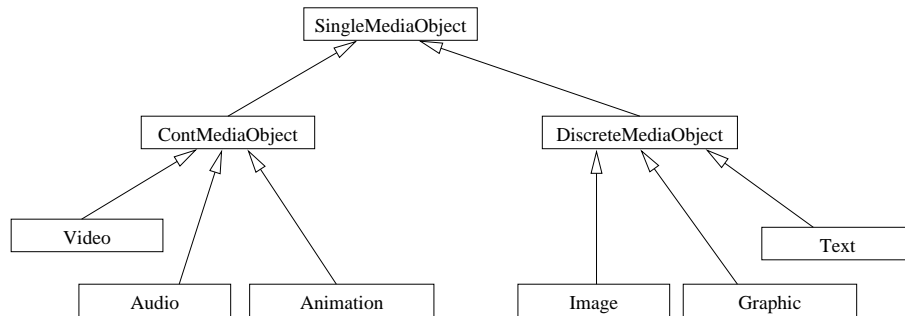


Fig. 2. Hierarchy of Media Objects

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.

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 employed. 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.

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.

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

Table 3. Services

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.

4.2 Media Integration Service

The Media Integration Service manages relations between media and data objects. The relations must be identified by a user and stored persistently. They can be defined in different ways. Here, a relation is defined by a name, a class in the FDB schema, the cardinality (1 or n), and the media type. A data object of the given class, selected by its OID, can reference through this relation one or more media objects of the given type, identified by their media-object identifiers. Based on these relations a virtual schema can be presented to the user.

This schema and the Media Search Service can be queried together in one query language. Users are thus provided with the option for fuzzy queries over media objects, too.

The Media Integration Services can also be used to store relations between single media objects i.e. multi-media objects or hyperlinks. A multi-media object describes the spatial arrangement and timing of single media objects. Hyperlinks represent arbitrary connections between media objects.

5 Architectures

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.

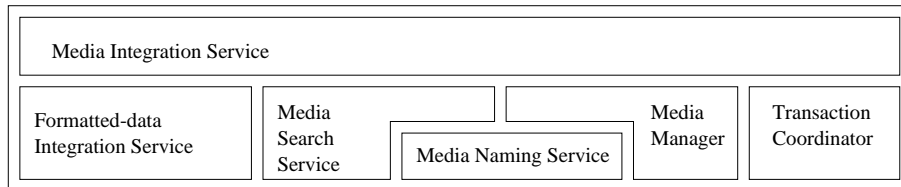


Fig. 3. Full DMDBMS

6 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 [7, 8]. 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 [12, 16, 1, 10, 14, 15, 9, 17].

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

HERMES [18] and TSIMMIS [6] 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 [19]. The federated object-oriented database management system contains objects which describe multimedia documents. An information object contains data that describe content, structure, and synchronization aspects, 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.

[13] 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 [13] 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.

7 Next Steps

This paper describes work in progress. The specification of the Media Integration Service will be completed. A query language which also supports search based on information retrieval is currently examined. This query language is based on the OQL from ODMG 2.0. A compiler for that language which decomposes queries into three parts (for the FDB schema, the media-data relations, 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.

Later the Media Integration Service will be enhanced to store and manage presentations and hypermedia, too, as mentioned in Sect. 4.2.

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. Sudarshan Chawathe et al. The tsmmis project: Integration of heterogeneous information sources. In *Proc. of IPSJ Conference*, 1994.
7. 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>.
8. Asuman Dogac et al. METU Interoperable Database System. www, 1995. <http://www.srdc.metu.edu.tr/mind/publications.html>.
9. D.James Gemmell et al. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, pages 40–49, May 1995.
10. 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>.
11. 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.
12. Ulrich Marder and Günter Robbert. The kangaroo project. In *Proceedings of the Third Int. Workshop on Multimedia Information Systems*, 1997.
13. Yoshifumi Masunaga. Multimedia Databases: A formal framework. In *Proceedings of the IEEE Computer Society Symposium on Office Automation*, pages 36–45, 1987.
14. A. Desai Narasimhalu. Multimedia Databases. *Multimedia Systems*, 4(5):226–249, 1996.
15. 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.
16. Thomas C. Rakow, Wolfgang Klas, and Erich J. Neuhold. Research on Multimedia Database Systems at GMD-IPSI. *IEEE Multimedia Newsletter*, 4(1), 1996.
17. 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.
18. 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>.
19. 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>.