A Mediator for Multimedia Systems

Joachim Biskup, Jürgen Freitag, Yücel Karabulut, Barbara Sprick
Fachbereich Informatik
Universität Dortmund
D-44 221 Dortmund
Germany

{ biskup | jf | karabulu | sprick } @ ls6.informatik.uni-dortmund.de

A multimedia system can be considered as a distributed, federated, extensible information system that supports a wide range of object types and allows a high degree of autonomy of its components. A multimedia mediator aims at providing a well-structured and controlled gateway to some application dependent part of the whole system. We present our overall design of such a mediator. The design is based on multimedia schemas and on object-oriented concepts. In particular we employ proxy objects for external multimedia items, and we introduce types for semi-structured multimedia items. High level query processing hides all details of external communications, and query optimization takes advantage of parallel processing capacities of external sources and of materializations of previous query answers in the proxy state.

1 Introduction and survey

Design, implementation and administration of multimedia systems require to combine the best available technologies from various fields of information processing, including knowledge represention, databases, information retrieval, image processing, computer graphics, and federated systems. Therefore, we should explore, which features taken from the various fields can be sensibly and pragmatically put together, in order to design useful and manageable components, the implementations of which can be administrated as an interoperable part of a large network based multimedia system.

Our *Multimedia Mediator* project aims at proposing just one such mixture. The project adopts the paradigm of a *mediator* sitting between a user layer, formed by a semantically rich knowledge representation system, and a possibly distributed collection of multimedia sources, which are presumably highly parallell external servers. The mediator should take advantage of traditional database technology as far as possible, in order to achieve a high degree of conceptual clarity and implementational efficiency, even for a large amount of data.

In order to do so the mediator is based on using *multimedia schemas* and on *object-oriented concepts.* The former extend the successful notion of a database schema and support well-structured application dependant instantiations, and the latter allow to maintain a representing object for each multimedia item visible in the mediator´s view. If such an item is actually stored externally, then the representing object is used as a *proxy object*. Proxy objects, though introducing some indirection, can be exploited in various ways to improve the overall efficiency.

Hence our approach emphasizes to employ the mediator as a generic tool that can be be fully instantiated for any appropriate application: on the descriptional level by a multimedia schema, and on the instantiational level by the proxy objects. The instantiations are intended to capture exactly the application dependant specifications, whereas all other features are transparently provided by the generic tool.

In this paper, we present the overall design of our multimedia mediator. Our presentation is organized in a top-down style. Section 2 reviews basic objectives, summarizes the general approach and outlines the architecture. Section 3 highlights query evaluation and optimization, and Section 4 introduces our treatment of semi-structured multimedia items. The topics of Section 3 and of Section 4 are exposed in more detail in a companion paper [7]. Finally, Section 5 briefly compares our design with alternative approaches and presents conclusions for current and future work.

2 Basic objectives and general approach for a multimedia mediator

2.1 Basic objectives

In our context a multimedia system can be considered as a distributed, federated, extensible information system that supports a wide range of object types and allows a high degree of autonomy of its components. Refining this view, we state ten basic objectives for *multimedia mediators*.

A mediator should

1) provide an application driven view on the whole multimedia system,
2) represent the specific knowledge pertinent to the application,
3) extract and maintain semantic constraints on the stored information,
4) supervise security constraints,
5) offer a multimedia query and manipulation language that is both expressive and efficient,
6) present multimedia query results,
7) make the whole system transparent by hiding all the subtle details of communications,
8) provide transparent translations among different data models as well as appropriate protocols for interactions, as required by the specific application,
9) integrate the various types required for the specific application within a uniform type system which includes appropriate types for semi-structured data,
10) easily adapt any extension to the federation structure or to the type system.

2.2 General approach and architecture

Our general approach for a multimedia mediator is based on two established technologies for database systems:

- using *multimedia schemas* as a self-description of a specific application, in order to achieve well-structured instantiations and control on the one side, and

- employing *object-oriented concepts*, in order to federate heterogenous, distributed and evolving information sources on the other side.

The general architecture is visualized by a diagram given in the Appendix. Selected features are explained in the following text.

2.2.1 Multimedia schemas

A multimedia schema is layered as sketched in the diagram.

The *view layer* of the multimedia schema offers (at least) two interfaces. The main *application view* is provided for the local users running the specific multimedia application, and, accordingly, it supplies the full functionality of the mediator. An additional *federation view* is intended for external users and typically supplies only restricted access to the mediator´s services.

The *conceptual layer* of the multimedia schema constitutes the instantiation of the mediator for the specific multimedia application. It contains a full structural and operational description of the multimedia application, as seen from the local users, and comprises in particular:

- multimedia *types*, for abstractly structuring the information managed by the mediator, and for manipulating that information;
- *conceptual classes* as abstract containers for multimedia objects sharing a common type, for inserting and deleting multimedia objects, for declaring one or several *search subtypes* for fast retrieval, for iterating over collections, for defining domains for searching, semantic constraints and security constraints, and for declaring *presentation layouts* of its members;
- *semantic constraints*, for maintaining application dependent invariants under update operations;
- *security constraints*, for restricting and controlling the information flow and for providing nonrepudable evidence of using the mediator;
- a *metaschema* as abstract knowledge representation tool, including semantic nets and linguistic dictionaries, for capturing the real world semantics of the multimedia application;
- generic *view classes*, for declaring presentation layouts and for enabling users to store query results as materialized views.

The *internal layer* of the multimedia schema provides the details of how the conceptual items are implemented. Most importantly, it comprises

- *site directions*, for indicating whether the conceptual multimedia item is locally available and, if not, for providing *communication protocols* for accessing that item elsewhere in the multimedia system.

Besides that innovative peculiarity, the internal layer refines the conceptual items as required by the overall three-layer architecture. In particular it comprises:

- data formats and procedures for implementing types,
- access structures and storage management for implementing classes, in particular for the search subtypes of the class,
- presentation details for class members,
- monitors for ensuring semantic constraints, and

- filters and audits for enforcing security constraints.

Finally, the remaining two nonstandard components of our conceptual schema must be refined:

- implementation details for the metaschema, and
- presentation details and refreshment strategies for view classes.

2.2.2 Object-oriented concepts

First of all, we employ object-oriented concepts as in any traditional object-oriented information system (see the discussion of multimedia schemas above).

Besides that, we take advantage of object-orientation for hiding the heterogeneity and distribution of the information sources within the whole multimedia system:

- Firstly, any multimedia item visible in the mediator's view, whether locally stored or only remotely accessible, is represented by a uniquely identified object that is under the control of the mediator. If the represented item is locally stored, we call that object a *local object*; otherwise, if it is externally stored, that object is used as a *proxy object*. We also allow that a single object is used in a mixed way, jointly representing locally and externally stored aspects.

Thus the mediator maintains a full *proxy state* of the application driven view of the whole multimedia system, which consists of all created objects of any kind. Accordingly, a user of the mediator gets the illusion of dealing with just this proxy state and he is not at all aware of the storage sites.

- Secondly, the class specific procedures for types hide all the subtle details on how to actually access the items represented by the local and proxy objects.

Again, a user of the mediator is not at all aware about the actual access methods.

2.3 Interfaces

From the user´s point of view the multimedia mediator appears nearly like an ODMG compliant database. The *query and manipulation interface* contains a sublanguage of the ODMG language which has been enriched by constructs for dealing with semi-structured multimedia items.

The *communication interface* handles protocols that are defined in the internal layer of the schema. These protocols are triggered by the class specific procedures when external sources must be accessed. Two features must be carefully designed: set-at-a-time execution and synchronisation among the mediator´s processes and the interacting external processes.

2.4 An example application: high performance multimedia server

Our multimedia mediator is designed to serve as an integration tool within an ambitious project for building a high performance multimedia query server, HPQS. The overall goal of the HPQS is to provide high-level, user-friendly, and interactive access to an application domain. In order to achieve high performance, the HPQS is based on a specialized parallel computing system which can be tailored to the most frequently required elementary search requests on texts and images of an application. In principle, there can be more than just one underlying specialized parallel system, and these systems can be remote from the user site. Therefore, given a semantically meaningfull query, our multimedia mediator has to extract appropriate sets of elementary subtasks to be processed by the possibly external parallel systems. It is important to note that the mediator has to forward *sets* of elementary subtasks which are *as large as possible* in order to take advantage of the parallel processing capacities of the underlying parallel systems.

On the one hand, we assume a massive amount of data, e. g. texts and images, processed by underlying parallel systems, on the other hand, we recognize semantically elementary but computationally complex search requests on texts and images as a dominant cost factor for query evaluation. Thus, we designed our multimedia mediator to employ not only *parallelism* but also *materialization* of query results, in order to avoid recomputations of elementary requests whenever possible and useful.

3 Query evaluation and optimization

For any *attribute value* we distinguish three cases: either it is locally stored and thus can be simply fetched, or it must be determined from external items by means of a communication, or it is redundantly available in both forms. Basically, a procedure for *reading* an attribute value first tries to find the value locally and if this attempt fails, it generates an appropriate command to query the external source.

When evaluating an *OQL query*, we have to call the procedures for reading attribute values. There are two simple brute force approaches to handle the calls of these procedures.

In the *one-pass approach*, the original OQL query is processed nearly as usual. However, whenever a call of a reading procedure starts an external communication, the query process is suspended until the external value has been supplied. As a side effect, after a call of the reading procedure the (real) attribute value is always locally materialized and can be made persistent.

In the *two-pass approach*, the original OQL query is processed by a four step transaction:

1) The original query *Q* is syntactically analyzed to determine which calls for reading procedures are required. Based on this analysis, a new transformed OQL query *Q_external* is generated. As a query result, it just delivers the set of all parameters to be passed to some function *communicate* later on, and finally to be forwarded as commands to the external sources.
2) The transformed query *Q_external* is actually executed.
3) Its output is appropriately sent to the external sources. Afterwards, the returning answers, containing the attribute values, are interpreted as insert commands that materialize the attribute values locally. Thus, when all answers have been returned, then all attribute values required by the original query are locally available.
4) Finally the original query *Q* is executed as usual.

The one-pass approach treats all external communications sequentially, and thus it is unlikely to be efficient. The two-pass approach treats all external communications simultaneously, and thus it can take advantage of the parallel computing power of the external servers. But it has other drawbacks. It does not allow parallelism between local processing and external communication. It cannot take advantage of any traditional optimization technique to reduce the number of attribute accesses. And it only works for a restricted kind of queries that allow to precompute the set of all objects for which an attribute value must be determined. Therefore, the *query optimizer* of the mediator is designed to select an appropriate optimization strategy for any given query. It has to decompose the query and to determine a corresponding optimal number of passes with an emphasis of avoiding the drawbacks mentioned above. In [7] we outline such "multiple two-pass" evaluations .

Any evaluation strategy potentially supplies some attribute values that have not been locally available before. This offers the option to materialize these newly available values permanently in the local storage. If they are subsequently needed for further queries they can be simply fetched, and thereby their possibly time and resource consuming recomputation can be avoided. Pursuing this option, we have to deal with *materialization* strategies and assumptions on *incomplete knowledge.* These topics are decided on the basis of the type of an attribute, as described in some more detail in [7].

Persistent materialization of attribute values offers the option that the multimedia mediator uses its proxy state to *cumulatively learn* more and more aspects of its application field. This option is particularly powerful in combination with semi-structured attributes: learning may enhance the knowledge about both the *structure* and the *content* of multimedia items.

4 Dealing with semi-structured multimedia items

In traditional database management systems we can achieve a clear separation of raw data, the time-varying instance, and its structural description, the time-independent schema. If data is only *semi-structured*, then, at most, only a part of its structural description can be time-independently declared in the schema. But, additionally, we propose that any single data unit also individually carries the description of its particular structure on the instance level. Then the particular structures on the instance level have to conform to the partial structural description on the schema level. Thus the partial structural description on the schema level is taken as an invariant to be maintained under update operations.

Extending the usual type systems, we introduce a new type constructor **self_describe** with two parameters which is used for multimedia items that are both semi-structured and self-describing. The first parameter must be a usual type, and it specifies the time-independent minimal constraints on the structure of allowed multimedia items. The second parameter must be a *meta-type* the values of which are types. It specifies which types are allowed as a time-varying self-description for the multimedia items that are allowed by the first parameter. The first parameter, a type, and the second parameter, a meta-type, must be *compatible* in the sense that the first parameter can be embedded in all types given by the second parameter. Here, type σ is *embedded* in type τ roughly means that the values of type σ can be understood as values of type τ, and that values of type τ can be queried as if they were of type σ. The extended type system has been introduced in [7] and is currently investigated in more depth.

The extended type system can be exploited in several ways. First, also the **self_describe** types are used as invariants to be maintained under updates, and all queries must be well-typed. Secondly, being data like other data, the self-description component of a value of a **self_describe** type can be queried at run-time. Thirdly, such a self-description component can be improved at run-time by analyzing a given multimedia item by appropriate procedures. Fourthly, if a query returns a set of values of some **self_describe** type, then we can determine the less refined type that can be embedded in all types occuring in the actual result set, i.e. we can attempt to compute the best available common description of all items in the query result.

5 Conclusions and future work

The paradigm of a mediator has first been advocated by G. Wiederhold , and since then it has attracted many researchers [16,18]. Our specific choices emphasizes to (re)use database technology as far as possible and useful.

Taking advantage of database technology for multimedia applications has also been explored by several projects, see for instance [2,5,8,11,12,13,14,15]. Our emphasis is on a mediator architecture. The mediator maintains an object oriented proxy state that represents a partially materialized view on possibly heterogenous and external multimedia sources, featuring both transparent and optimized set oriented access to external sources.

There are already many studies and proposal on dealing with semi-structured data, see [1] for a recent survey. Our particular proposal for types, which includes the new type constructor **self_describe,** combines aspects of previously discussed multimedia types such as in [15], of structuring schemas and tagged union types [2,3], and of graph schemas [9].

Since high efficiency is of great importance, also multimedia query optimization has been studied under various aspects, see for instance [2,4,6,10,17]. One special interest for our multimedia mediator has been to exploit the potentials of parallelism of external agents and its tradeoff with materialization. In contrast to [4], we take advantage of the proxy state, i.e. of information on the instance level, rather than of so-called semantic invariants, i.e. information on the conceptual level.

Our design of the multimedia mediator is currently started to be implemented in a prototype fashion. The prototype is intended to be used within the HPQS project. The implementation tries to use generally available systems as far as possible. For the database functionality we base the mediator on the O_2 system. For the communication functionality we base the mediator on the CORBA proposal. Basically, the two functionalities are combined by procedures for reading attribute values, as required by the one-pass approach or the approaches with several passes, respectively, together with the function *communicate.*

There are many avenues for future work. Most importantly, our design of a multimedia mediator has to be evaluated and tuned with respect to performance characteristics. The impact of maintaining a proxy state should be determined also experimentally, in particular the tradeoff between selectivity, which is effected by the materialization strategies, and parallelism of the external agents. Conceptually, query optimization should be studied in more detail, in particular taking into consideration the impact of the extended type system. A depth-in study of a our proposal of the type constructor for semi-structured data would also be worthwhile. Finally, there are two further great challenges, namely transforming conceptually declared semantic constraints and security constraints into effective and efficient monitors and filters.

6 References

[1] S. Abiteboul, Querying semi-structured data, Proceedings *6th International Conference on Database Theory*, ICDT ´97, Delphi, Greece, Lecture Notes in Computer Science 1186, Springer, Berlin etc., 1997, pp. 1 - 18.

[2] S. Abiteboul, S.Cluet, V. Christophides, T. Milo, G. Moerkotte, J. Siméon, Querying documents in object databases, Journal of Digital Libraries , to appear.

[3] S. Abiteboul, D. Quass, J.McHugh, J. Widom, J.L. Wiener, The Lorel query language for semistructured data, Journal of Digital Libraries , to appear.

[4] S. Adali, K.S. Candan, Y. Papakonstantinou, V.S. Subramahnian, Query caching and optimization in distributed mediator systems, Proceeding *1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, SIGMOD Record (1996) 25:2, pp. 137 - 148.

[5] M. Adiba, STORM: an object-oriented multimedia dbms, In: *Multimedia Database Systems* (eds: K.C. Nwosu, B. Thuraisingham, P.B. Berra), Kluwer, Boston etc., 1996, pp. 47 - 88.

[6] E. Bertino, F. Rabitti, S. Gibbs, Query processing in a multimedia document system, ACM Transactions on Office Information Systems (1988) 6:1, pp.1 - 41.

[7] J. Biskup, J. Freitag, Y. Karabulut, B. Sprick, Query evaluation in an object-oriented multmedia mediator, Proceedings *4th Int. Conference on Object-Oriented Information Systems*, Brisbane, Australia, Nov. 1997, Lecture Notes in Computer Science, Springer, Berlin etc., to appear.

[8] J.A. Blakeley, Data access for the masses through OLE DB, Proceeding *1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, SIGMOD Record (1996) 25:2, pp. 161 - 172.

[9] P. Buneman, S. Davidson, M. Fernandez, D. Suciu, Adding structure to unstructured data, Proceedings *6th International Conference on Database Theory*, ICDT ´97, Delphi, Greece, Lecture Notes in Computer Science 1186, Springer, Berlin etc., 1997, pp. 336 - 350.

[10] S. Chaudhuri, L. Gravano, Optimizing queries over multimedia repositories, Proceeding *1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, SIGMOD Record (1996) 25:2, pp. 91 - 102.

[11] C.Y.R. Chen, D.S. Meliksetian, M. Cheng-Sheng Chang, L.J. Liu, Design of a multimedia object oriented dbms, Multimedia Systems (1995) 3, pp.217 - 227.

[12] N. Fuhr, Object-oriented and database concepts for the design of networked information retrieval systems, Proceedings *5th International Conference on Information and Knowledge Management*, 1996, pp. 164 - 172..

[13] K. Groenbaek, J.A. Hem, O.L. Madsen, L. Sloth, Cooperative hypermedia systems: a Dexter-based architecture, Communications of the ACM (1994) 37:2, pp. 64 - 74.

[14] W. O´Connell et al, A teradata content-based multimedia object manager for massively parallel architectures, Proceeding *1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, SIGMOD Record (1996) 25:2, pp. 68 - 78..

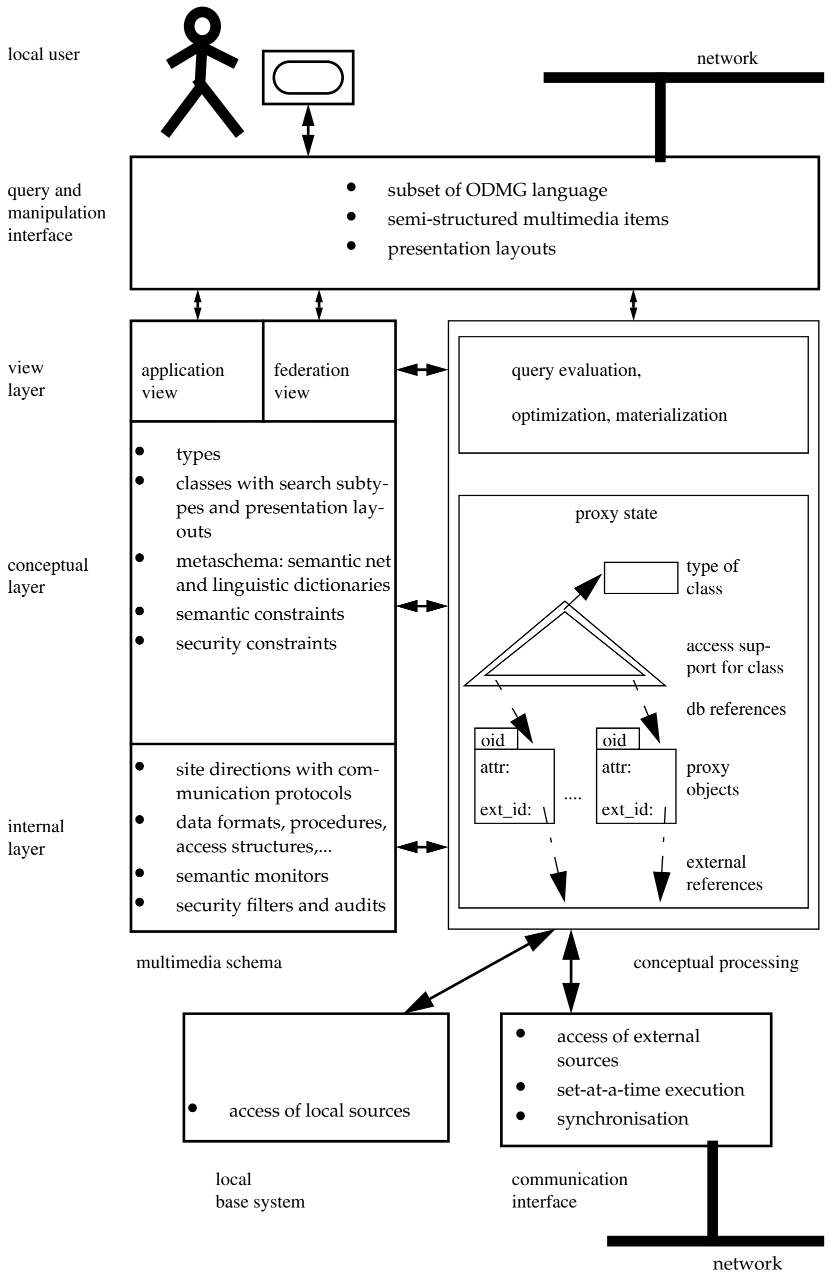
[15] M.T. Özsu, D. Szafron, G. El-Medani, C. Vittal, An object-oriented multimedia database system for a news-on-demand application, Multimedia Systems (1995) 3, pp. 183 - 203.

[16] V.S. Subrahmanian, HERMES: a heterogeneous reasoning and mediator system, submitted for publication (see http//www.cs.umd.edu/hermes).

[17] J.D. Ullman, Information integration using logical views, Proceedings *6th International Conference on Database Theory*, ICDT ´97, Delphi, Greece, Lecture Notes in Computer Science 1186, Springer, Berlin etc., 1997, pp. 19 - 40.

[18] G. Wiederhold, Mediators in the architecture of future information systems, IEEE Computer (1992) 25:3, pp. 38 - 49.

7 Appendix



local user
network
query and
manipulation
interface
subset of ODMG language
semi-structured multimedia items
presentation layouts
view
layer
application
view
federation
view
query evaluation,
optimization, materialization
conceptual
layer
types
classes with search subtypes and presentation layouts
metaschema: semantic net and linguistic dictionaries
semantic constraints
security constraints
proxy state
type of
class
access support for class
db references
oid
attr:
ext_id:
....
proxy
objects
internal
layer
site directions with communication protocols
data formats, procedures, access structures,...
semantic monitors
security filters and audits
external
references
multimedia schema
conceptual processing
access of local sources
access of external sources
set-at-a-time execution
synchronisation
local
base system
communication
interface
network