VirtualMedia: Making Multimedia Database Systems Fit for World-wide Access[1]

Extended Abstract

Ulrich Marder
University of Kaiserslautern
Dept. of Computer Science
P. O. Box 3049
D-67653 Kaiserslautern
Germany
marder@informatik.uni-kl.de

1 Introduction

In the age of the world-wide web global media data—often called media assets—, stored in multimedia database management systems (MM-DBMS), will increasingly be made accessible to virtually everyone. Hence, lots of heterogeneous clients with different, not safely predictable capabilities of storing, processing, and presenting media data are willing to access these MM-DBMSs (which may be part of a digital library). In general, there exist two categories of client applications. The first only want to retrieve media objects for presentation or possibly printing. The second (additionally) create media objects or modify them through editing and/or composition. Assuming a sort of unbalance between these application types sounds reasonable: Usually there will be many applications of the presentation type and much less of the other type. Thus, one could be tempted to optimize the system(s) for presentation only. This would, however, most likely result in missing the much stronger quality and performance demands of media editing applications.

Consequently, media servers being an (essential) part of multimedia database management systems should provide physical data independence. Today's media servers—especially continuous media servers—, however, do not provide physical data independence at all. One—if not the main—reason for this is performance. Physical data independence *without optimization* undoubtedly costs a lot of performance. Therefore, this optimization problem (and solving it) is both initiator and quintessence of the VirtualMedia approach.

Obviously, we are facing considerable problems on attempting to provide physical data independence with a media server (or MM-DBMS, respectively): Such systems tend to require frequent format conversions inevitably resulting in bad performance. They may inadvertently lose data due to irreversible updates. Moreover, hiding the internal data representation from the client also means that all the strongly necessary optimization is to be accomplished by the server, which is both more difficult and more promising than leaving optimization to the applications. The following section summarizes some recent approaches to solve (some of) these problems. Section 3 briefly introduces VirtualMedia, followed by concluding remarks.

2 Related Work

Basically, our approach originates from an attempt to realize so-called *Media-specific Abstract Data Types* (MADT) [KMM94, MR97]. The goal of the MADT concept was to introduce new (DBMS-) data types for media objects that provide almost the same abstractions as traditional "built-in" data types. The most problematic abstraction this concept should support is *format independence*. Obviously, the key to format independence is reconciling the conflictive requirements regarding the internal and external raw data format. Since the internal format only depends on the DBMS, its properties should be optimal for storing and processing the data. This internal format, however, must also guarantee application neutrality, which means: It must not loose any bit of information provided by the creator of a media object. Such information loss could occur, e. g., by using a DCT-based compression

[1] This work is supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Sonderforschungsbereich (SFB) 501 "Development of Large Systems with Generic Methods".

scheme with the internal format. The external format, on the other hand, ideally depends only on the application. Hence, there might be as many different external formats as there are different applications.

While introducing desirable functional requirements, the MADT concept considerably complicates satisfying essential non-functional requirements. To make this obvious, assume that about 80% of the accesses to a media object refer to its original external format, whereas the remaining 20% involve some transformations resulting in a different external format. Thus, an obvious "simple" approach—using a fixed standard media format internally—would result in a high format conversion overhead. We could avoid this by always adopting the external format (at media object creation time) as the default internal format. This solution clearly supports loss-free storage of media objects. However, what about optimization of media processing operations, then? How could these MADT operations be implemented optimally, if the internal data format is only determinable at run-time?

The concept of *Enhanced Abstract Data Types* (E-ADT) [Ses98], which has been realized in the ORDBMS prototype *Predator*, provides a solution regarding the optimization of composite operations (transformation requests in our terminology). In particular, the E-ADT approach requires such transformation requests being specified (or interpreted, respectively) in a descriptive manner, thus enabling semantic optimization (e. g., permutation or replacement of operations). The salient feature of the E-ADT approach—its tight and elegant integration with traditional relational database technology—, however, also seems to enforce not considering more advanced optimization strategies like cross-media optimization or optimization-driven materialization. Further, neither format independence nor the irreversibility problem are addressed.

Within the AMOS project at GMD IPSI a concept called *presentation independence* has been developed [RKN96]. This abstraction aims at optimizing pre-orchestrated presentations for differently equipped clients. The QoS of such a presentation automatically adapts to the client's facilities at runtime. There are, however, no operations for ad-hoc creation or modification of presentations. Thus, the physical data independence provided is kind of "static" (besides being dedicated to presentation only).

Commercial ORDBMSs (available, e. g., from Informix, IBM, and Oracle) are extensible by defining and implementing *User-defined Types* (UDT). This mechanism is also extensively used to enhance those systems with media data types (for some examples see [Inf97]). While the vast majority of these media extensions do not provide physical data independence, two exceptions from this rule should be pointed out: (1) In [HSH+98] a continuous media DataBlade providing device independence, location transparence, and presentation independence is described, and (2) [WHK99] presents a DB2 Extender for images providing format independence where materialization is controlled through cost-based optimization.

3 The VirtualMedia Approach

Referring to the state-of-the-art outlined in the previous section, our goal is a framework supporting (1) different media types with rich functionality and physical data independence like in the MADT concept, (2) semantic optimization similar to the E-ADT approach, (3) cost-optimized materialization as in [WHK99], and (4) *dynamic* presentation independence.

There is a fifth requirement which is generally neglected in all ADT-based approaches: (5) support for optimizing media transformations. Regarding the growing heterogeneity of possible clients for world-wide accessible MM-DBMS (ranging from high-performance workstations connected to high-speed networks down to WAP-enabled mobile phones) this requirement becomes more and more important. Focusing on media transformations moves the operators (also known as *filters*) performing these transformations into the center of interest. To stay abreast of this focus shift we introduce the notion *transformation independence* meaning "physical data independence with respect to general media transformations (i. e., not only media ADTs)".

Transformation independence can be shortly characterized as a way of generally specifying the semantics of (arbitrarily complex) media transformations while abstracting from places of execution, execution sequences (of atomic operations), and persistence considerations (i. e., how, when, where, and how long to store media data in the database). Particularly, this means, transformation independence completely hides (to both the application and the MADT programmer), which physical media objects will be eventually materialized by the DBMS. Hence, the media objects visible to client appli-

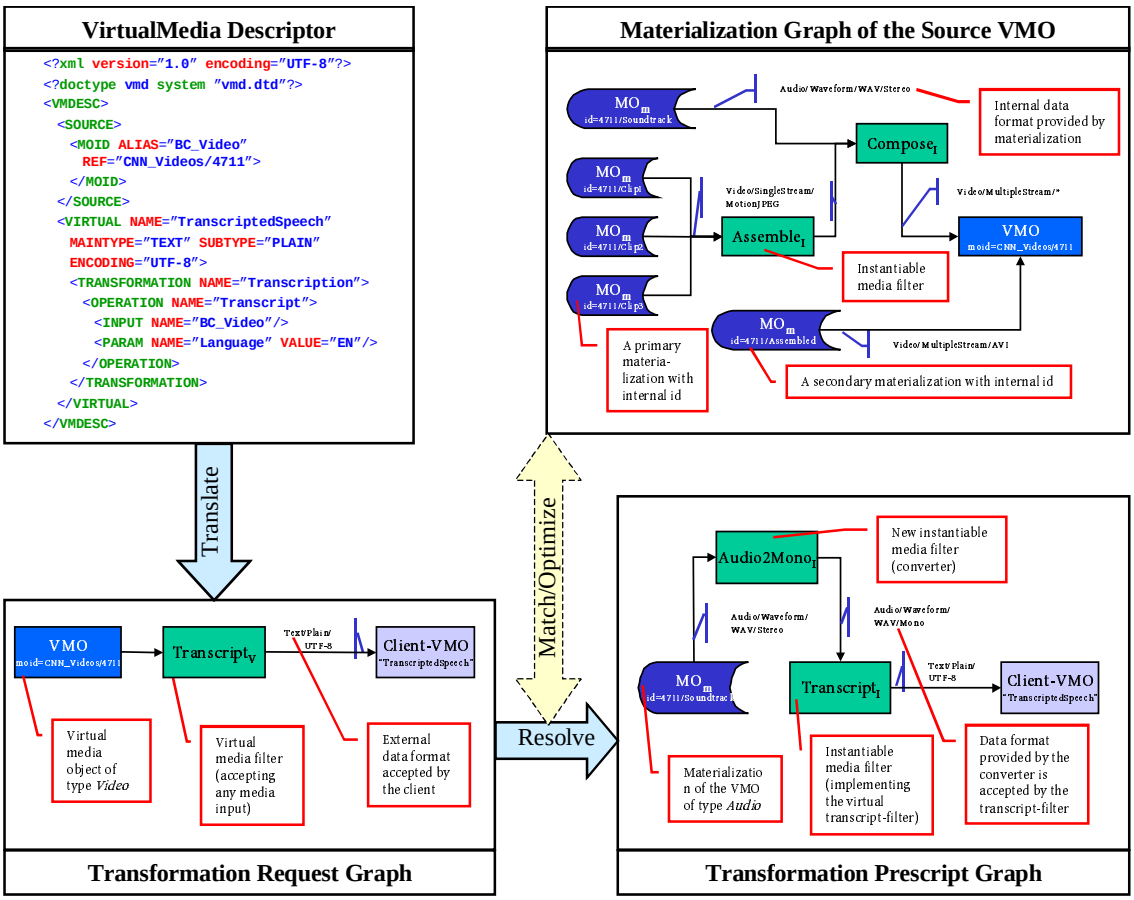

Figure 1: Transformation independence realized with virtual media objects (example)

cations are in fact *virtual* media objects. Operations on such virtual media objects (VMO) have to be mapped to (semantically equivalent) operations on the internally materialized objects. That means, the operations on VMOs are virtual themselves, hence the name *VirtualMedia*.

Thus, VirtualMedia has to provide the means for describing how virtual operations are to be applied to virtual media objects and how this can be mapped to real operations on real media objects. We use an example to illustrate the major aspects of the VirtualMedia concept (cf. Fig. 1): Assume a video object being stored in the database where the video's content is a talk given by a famous scientist. A client of the database is interested in this talk, but happens to be not equipped for video or audio presentation. Hence, the client requests a textual transcript of the talk for presentation on screen. Normally, the client would probably try to instruct the server to somehow isolate the audio part of the video and then pass it on to an appropriate transcription engine. Transformation independence, however, requires a different procedure: The semantics of the transformation are described in a *VirtualMedia descriptor* which is then passed to the server. The server translates the descriptor into a *transformation request graph*. This is a DAG where start nodes are source-VMOs, end nodes are client-VMOs, and intermediate nodes are virtual filters.

The transformation request graph has to be resolved yielding a semantically equivalent transformation prescript graph where all source-VMOs are replaced by qualified materializations, all virtual filters are replaced by appropriate instantiable filters, and no format mismatches occur. For this resolution process, the server needs materialization graphs of the source-VMOs, maps for looking up implementations of virtual filters, maps for finding format converters, and semantics-preserving rules for stepwise manipulation of filter graphs. Generally, there exist several different semantically equivalent prescript graphs. Hence, also a cost function is required to determine the cost-optimal[2] prescript graph.

[2] Other reasonable, but possibly counteracting optimization criteria are, e. g., media quality or delivery latency.

4 Conclusions

We claimed that in the age of the world-wide web MM-DBMSs should dynamically adapt to a heterogeneous and somewhat unpredictable “zoo” of client applications without loosing (much) performance. This requirement has been abstractly characterized as a generalization of data independence called *transformation independence*. The main idea of this abstraction is reducing the creation, retrieval, and modification of media objects to what can be called the "pure application semantics". The consequences are multiple optimization dimensions being left for exploitation by the server. The VirtualMedia concept realizes transformation independence based on virtual media objects being described and manipulated through filter graphs. With this concept, optimization is basically characterized as the process of optimally matching transformation request graphs and materialization graphs.

Current work focuses on determining algorithms, data structures, and cost functions that are applicable for the translation, resolution, and optimization of transformation requests. Another point of interest is the investigation of different architectural concepts, e. g., examining how object-relational DBMSs could be enhanced to support VirtualMedia.

References

[HSH+98] Hollfelder, S., Schmidt, F., Hemmje, M., Aberer, K., Steinmetz, A.: Transparent Integration of Continuous Media Support into a Multimedia DBMS. In: Proc. Int. Workshop on Issues and Applications of Database Technology (Berlin, Germany, July 6–9), 1998.

[Inf97] Informix Digital Media Solutions: The Emerging Industry Standard for Information Management. Informix White Paper, Informix Software, Inc., 1997.

[KMM94] Käckenhoff, R., Merten, D., Meyer-Wegener, K.: MOSS as Multimedia Object Server – Extended Summary. In: Steinmetz, R., (ed.): Multimedia: Advanced Teleservices and High Speed Communication Architectures, Proc. 2nd Int. Workshop IWACA '94, (Heidelberg, Sept. 26–28), Lecture Notes in Computer Science vol. 868, Berlin: Springer-Verlag, 1994, pp. 413–425.

[MR97] Marder, U., Robbert, G.: The KANGAROO Project. In: Proc. 3rd Int. Workshop on Multimedia Information Systems (Como, Italy, Sept. 25–27), 1997, pp. 154–158.

[RKN96] Rakow, T., Klas, W., Neuhold, E.: Abstractions for Multimedia Database Systems. In: Proc. 2nd Int. Workshop on Multimedia Information Systems (West Point, New York, USA, Sept. 26–28), 1996.

[Ses98] Seshadri, P.: Enhanced abstract data types in object-relational databases. In: The VLDB Journal Vol. 7 No. 3, Berlin, Heidelberg: Springer-Verlag, Aug. 1998, pp. 130–140.

[WHK99] Wagner, M., Holland, S., Kießling, W.: Towards Self-tuning Multimedia Delivery for Advanced Internet Services. In: Proc. 1st Int. Workshop on Multimedia Intelligent Storage and Retrieval Management (MISRM'99) in conjunction with ACM Multimedia Conference, Orlando, Florida, Oct. 1999.