James A. Hendler, Editor
University of Maryland
hendler@cs.umd.edu

Making peace with your multimedia

Sibel Adalı, Rensselaer Polytechnic Institute

As multimedia applications grow in feasibility and popularity, multimedia information management becomes an increasingly critical task. Many data-representation and processing models for multimedia have been proposed and implemented, and as these options diversify, communication between them becomes harder. Applications are forced to choose the right multimedia-management system to address a problem domain's specific needs, resulting in expensive systems that are hard to maintain and expand. The situation begs the question, "Is there, if at all, one unifying theory for mapping out the multimedia functions of an application in a declarative framework?" And, if so, "How and at which levels should we model interactions between separate multimedia information systems?"

To address these questions, I am constructing a new computing architecture called MMII (*M*ulti*m*edia *I*nformation *I*ntegration), which can operate with multiple multimedia information systems. MMII lets expressive, but possibly incompatible, information-processing methods coexist peacefully in a loosely coupled system. Like any integrated system, it allows the use of autonomous and dedicated systems for specific tasks. MMII also provides methods for integrating multimedia functionality at different levels of operation. To demonstrate the types of services provided by MMII, I use an interface called I.SEE (*I*ntegrated *Se*arch *E*ngine).

Defining the problem

A multimedia information system often brings together the following types of operations on top of multimedia objects:[1]

- A specific query language and engine that combines (1) attribute-based methods for cataloging and accessing individual objects and (2) function-based methods for performing classification operations on databases of objects;
- A presentation layer that combines the results of queries into multimedia presentations; and
- A delivery layer that delivers objects in the given presentation constraints.

Depending on the particular system, the delivery layer might also handle the user's interactions with the presentations and might invoke the query processor when necessary. Most often, the query and data-definition languages developed for such systems are application-specific, presenting a predefined bundle of the above multimedia operations. Developers do not create this bundle with the ability to extend its functionality to multiple systems. However, traditional data-management systems offer the user multiple views over the same information space. Moreover, a view is, in a sense, the customization of an information system for a specific user's needs.

In systems with a large array of options, customization should not be an afterthought, but should function as part of the core data-management services. MMII facilitates the integration of heterogeneous sources under a common umbrella, which is not just an amalgamation of multiple functions, but also an extensible and fully customizable multimedia information-management system.

MMII

The MMII architecture comprises three layers (see Figure 1) for the integration of heterogeneous information systems:

- *Query mapping*, for lower-level content and query-mapping mechanisms between individual systems;
- *Logical integration*, for higher-level logical integration of their services; and
- *Interactive presentation*, for a flexible presentation-management system.

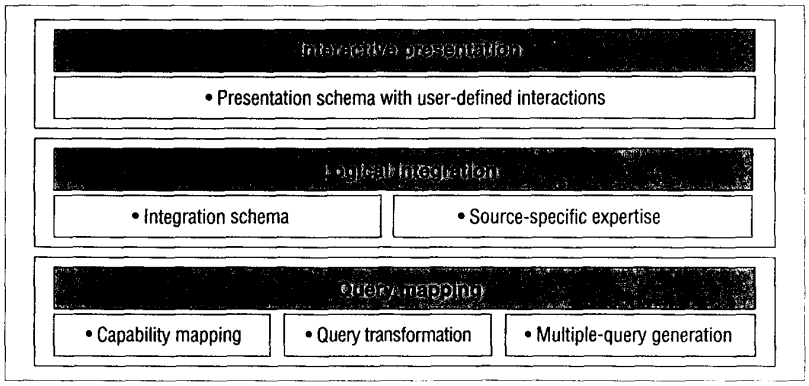

Figure 1. The Multimedia Information Integration architecture.

Each layer is programmed using a declarative specification language that is easy to build and customize. In fact, all three layers are natural extensions of well-known database-management concepts for defining compositional, logical, and operational aspects of information integration and management. My research group at the Multimedia Integration Lab has implemented the I.SEE interface (for more details, visit *http://www.cs.rpi.edu/research/isee*) as an initial test bed for this architecture.[2]

Query mapping. This layer is a gateway between multiple multimedia information systems with dedicated query interfaces. The user enters queries through either an existing interface (using all its specific options) or a new interface that contains an amalgamation of options from various interfaces (see Figure 2). The interface may query the sources using known attributes, such as "Find all pictures containing Mr. Smith or Mr. Jones." The query interface then looks up its knowledge base and finds out which sources can support which query-parameter combinations. This knowledge can be expressed using view definitions or a common ontology, as is the case in many information-integration systems.[3,4] This results in a set of alternate queries to distinct sources.

In addition, the interface may support queries with more of a multimedia flavor: "Find all pictures containing Mr. Smith wearing a hat and standing next to Mr. Jones." As a declarative query, the emphasis falls on whether the query can be answered, not how it will be answered. The query processor must find a query plan and the appropriate sources. The processor might run a text-based search on picture captions in one database or a spatial search on the stored locations of objects in another database. But, what if hat objects are not annotated in any of the databases? How and when should the query processor relax the query?

Most information-integration systems model multimedia queries such as the spatial query above as black boxes. This limits the use of these queries to those queries that are abstracted (wrapped) to the higher-level integration language.

Alternately, you could model all possible queries at the same level of complexity, supporting only the lowest common denominator. In this approach, if an integrated system is accessing multiple and incompatible information-retrieval engines, it will probably choose to support simple, keyword searches.

MMII's query-mapping service lets system programmers define exactly what can be supported by a specific source. The source description is in fact a parser, capable of processing queries with arbitrary structures. The capabilities are defined in terms of views that map a foreign information source to the user's known information source, which serves as the basis for any translation. You cannot map multimedia functions to a universal model without an *anchor*, a set of known operations. However, given the appropriate basis, you can specialize a general system of mappings to include only operations that are interesting to a user.

Query-mapping services provide a declarative framework based on Church-Rosser string-rewriting systems,[5] in which

- a parser characterizes a set of valid structured user queries containing attribute-based selection criteria and functional components from an arbitrary input query language;
- a verification service for different parsers ensures deterministic behavior and efficient execution; and
- an independent weight mechanism lets the query-mapping layer use one or more metrics for query relaxation and transformation. (These metrics can be defined among different query parameters or even among query segments. The query-mapping algorithm remains unchanged as query-transformation metrics change from system to system.)

This mapping framework provides a new mechanism for transforming and translating complicated queries, hence pushing more involved information-processing operations to the dedicated sources as much as possible. For example, I.SEE combines various query capabilities of multiple search engines. When necessary, an over-specified user query is relaxed using the above techniques to access a spe-

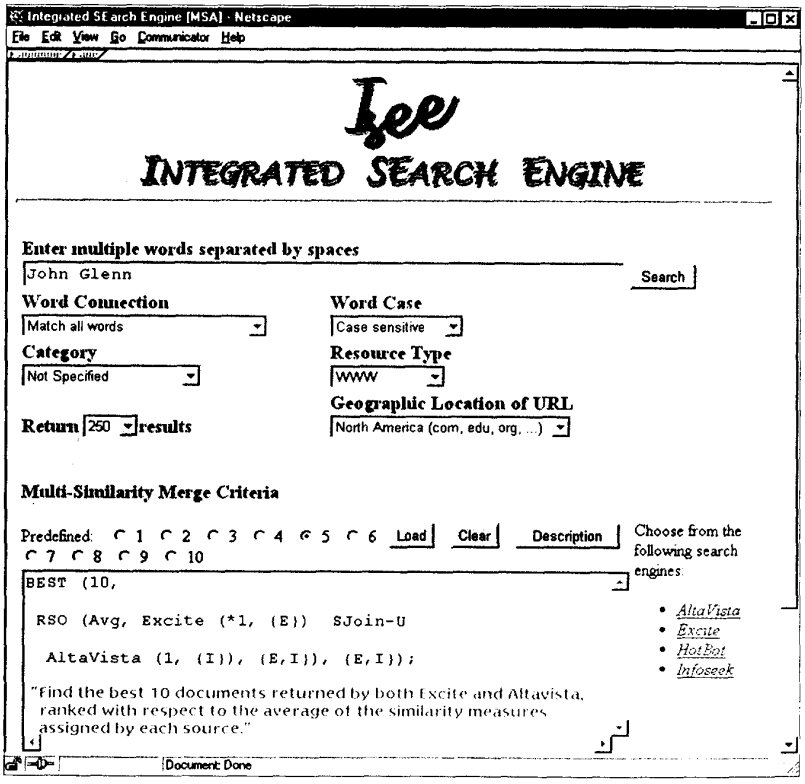

Figure 2. The I.SEE query interface, with which users can formulate a query to gather information from multiple sources.

cific source. I.SEE also combines selection operations on relations, such as the geographic location of objects or categories, with functional components, such as keyword-based classifications combined with different connectives.

I.SEE lets users choose the query methods. The underlying mapping system suffices to compute the invocation methods of the sources for the queries.

Logical integration. Because the query-mapping layer makes it possible to define associations between multimedia query methods supported by different information sources, a user's query is typically mapped to multiple sources. Additionally, the sources sometimes rank the results they return with respect to a similarity criteria. However, an integrated engine might use additional ranking criteria between different sources, depending on the query, the sources' reliability, and the translation of the query to different query interfaces. The final ranking of answers might also depend on other issues such as the amount of consensus or divergence among the sources.

Users should receive results in one coherent view. We call this view an *integration schema*, which we define using a declarative language called the multisimilarity *algebra*.[6] This algebra identifies each different query for a specific source through a unique type identifier. The multisimilarity algebra provides operators that resemble the relational algebra operators for the manipulation of type identifiers and ranked information. It links the results of multimedia operations with relational queries, allowing a multimedia query engine to be embedded in a relational framework as well. The declarative language also facilitates the development of query rewriting and optimization methods. This framework lets users specify whether they favor either results from sources that can support the most specific translation of their query or results returned by many sources.

The multisimilarity algebra expressions specify the logical integration of different queries as templates in a schema. After the translation phase, MMII chooses the most appropriate template for each query and integrates the results accordingly. These templates can be predefined or acquired using feedback from different sources. Because the query-mapping layer parses a query into small-unit functions, the integration layer can accumulate expertise based on these unit operations. Feedback can come from users, based on their preferences, or from classification engines, based on the effectiveness of different input parameters. MMII can use this to derive weights to fine-tune the integration schema defined as an expression in the multisimilarity algebra.

I.SEE features a complete implementation of the multisimilarity algebra. I am working on automating the use of different algebraic expressions by taking into account how much queries are relaxed with respect to different search engines. This would enable compensation for the information loss incurred during query mapping, by adjusting the integration schema.

Interactive presentation. The final step to any query-processing system is result presentation (see Figure 3). The presentation involves the interplay of two components: formatting the results for browsing and interactively delivering them. Formatting requires assigning spatial and temporal attributes to the results—deciding when and where different objects should appear. In addition, formatting links different presentations using logical user interactions. Answers can be organized into disjoint presentations, which can in turn be combined logically by interactions to deliver what the user wants to see. For example, a user might want to combine nine sources in three distinct groups using different integration schema, and choose the format of a window with three buttons, each of which activates the presentation for a specific group.

Delivery is based on the time-critical components of a query. Delivery ensures that objects can be ready before the user requests them. Query processing in a multimedia information-integration framework opens up many new options for the presentation of results, some of which I have explored in detail.[7] An interactive presentation framework is the perfect method for supporting a large array of such query methods.

For example, suppose we express query requests by `query`, `min`, `max`, and `sort_`

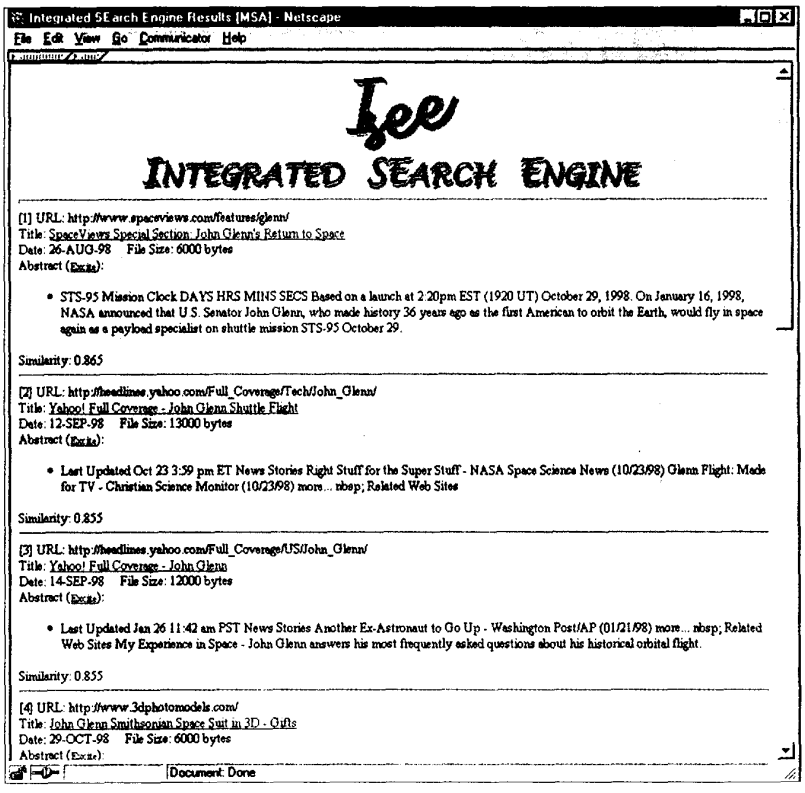

Figure 3. A sample hyperlinked presentation of results returned by I.SEE.

Sibel Adalı is an assistant professor in the Computer Science Department at the Rensselaer Polytechnic Institute. Her Multimedia Integration Lab conducts research on flexible frameworks for indexing and querying of multimedia information sources. Her research team has implemented several research prototypes that facilitate the integration of query and data-processing capabilities of multiple software packages. Her research interests lie in database-interoperability issues and query optimization in heterogeneous and distributed information systems. She received her BS in computer engineering and information services from Bilkent University, Ankara, Turkey, and her MS and PhD in computer science from the University of Maryland. Contact her at the Computer Science Dept., Rensselaer Polytechnic Inst., Troy, NY 12180; sibel@cs.rpi.edu; http://www.cs.rpi.edu/~sibel.

`type`. The MMII query engine will present the user the first `min` objects in the result to `query` as soon as they are available. At this point, the engine lets the user interact with the answers, and the users can retrieve the newly computed objects to the query until the engine finds a total of `max` objects. This lets the user view results even if the query is still executing. Moreover, `sort_type` describes additional criteria with respect to the partial ranking of objects—whether or not the ranking is allowed. If partial ranking is allowed, the query engine does not wait until it determines the final object ranking before presenting the objects to the user. This delivery method encompasses simple user interaction—namely, getting the next set of results to a user's query. Many other query-processing methods might prove meaningful in distributed, heterogeneous, and multimedia information platforms.

In general, you can define a large number of presentations based on the existing presentations, objects, and user-defined interactions. These options are defined by a template that shows how you can obtain presentations from the existing components. The *presentation schema* is a collection of such templates that MMII chooses for a given query. This system is based on a declarative multimedia presentation language that allows querying and authoring of interactive multimedia presentations.[8] It also allows the presentation of results in many different ways based on user profiles.

The MMII architecture concentrates on the information-processing aspects of multimedia information systems. It lets me describe the capabilities of information sources at multiple levels and choose the best tools for the application at hand. In addition, it lets me talk about the information-processing characteristics of an application—independent of its tools. In such a system, I can organize my application into smaller and reusable components, and easily migrate my solutions to newer and better tools. All pieces of the MMII infrastructure are being implemented as stand-alone research prototypes. The flagship application, I.SEE, is the first test bed for experimentation with the interplay of different layers.

All pieces of the MMII infrastructure are being implemented as stand-alone research prototypes. The flagship application, I.SEE, is the first test bed for experimentation with the interplay of different layers.

MMII opens up many exciting research areas. The interaction between query-mapping and logical-integration layers can be used to accumulate expertise about different multimedia systems. Instead of working on the best-possible classification method for various applications, I can learn when a specific classifier works well. I can improve the classifiers by letting them specialize in these cases. The interaction between the logical-integration and answer-presentation layers can be used to tailor an educational application's behavior. The application can adapt itself to the learning pattern and learning speed of different students. This way, we can separate the educational content from the pedagogical methods, creating customized learning environments on the fly to answer individual students' needs.

Multimedia information management provides us with a new degree of freedom in query processing. MMII is a starting point in my investigation on how we can take full advantage of this freedom. ■

References

1. V.S. Subrahmanian, *Principles of Multimedia Database Systems*, Morgan-Kaufmann, San Francisco, 1998.
2. S. Adalı, C. Bufi, and Y. Temtanapat, "Integrated Search Engine," *Proc. KDEX '97 1997 IEEE Knowledge and Data Eng. Exchange Workshop*, IEEE Press, Piscataway, N.J., 1997, pp. 140–147.
3. Y. Arens et al., "Retrieving and Integrating Data from Multiple Information Sources," *Int'l J. Intelligent and Cooperative Information Systems*, Vol. 2, No. 2, 1993, pp. 127-158.
4. A. Levy, D. Srivastava, and T. Kirk, "Data Model and Query Evaluation in Global Information Systems," *J. Intelligent Information Systems*, Vol. 5, No. 2, 1995, pp. 121–143.
5. S. Adalı and C. Bufi, "A Flexible Architecture for Query Integration and Mapping," *Proc. CoopIS '98: Third IFCIS Conf. Cooperative Information Systems*, IEEE Computer Society Press, Los Alamitos, Calif., 1998, pp. 341–351.
6. S. Adalı et al., "A MultiSimilarity Algebra," *Proc. 1998 Sigmod Conf. Management of Data*, ACM Press, New York, pp. 402–413.
7. S. Adalı et al., "Query Caching and Optimization in Distributed Mediator Systems," *Proc. 1996 Sigmod Conf. Management of Data*, ACM Press, New York, pp. 137–148.
8. S. Adalı, M.L. Sapino, and V.S. Subrahmanian, *Interactive Multimedia Presentation Databases, I: Algebra and Query Equivalences*, Tech. Report 98-04, Computer Science Dept., Rensselaer Polytechnic Inst., Troy, N.Y., 1998.