Integrating and Accessing Heterogeneous Information Sources in TSIMMIS*

Extended Abstract

Hector Garcia-Molina, Joachim Hammer, Kelly Ireland,
Yannis Papakonstantinou, Jeffrey Ullman, Jennifer Widom

Department of Computer Science
Stanford University
Stanford, CA 94305-2140

last-name@cs.stanford.edu

1 TSIMMIS and its Components

A common problem facing many organizations today is that of multiple, disparate information sources and repositories, including databases, object stores, knowledge bases, file systems, digital libraries, information retrieval systems, and electronic mail systems. Decision makers often need information from multiple sources, but are unable to get and fuse the required information in a timely fashion due to the difficulties of accessing the different systems, and due to the fact that the information obtained can be inconsistent and contradictory.

The goal of the *TSIMMIS*[1] project is to provide tools for accessing, in an integrated fashion, multiple information sources. Numerous other recent projects have similar goals and are discussed briefly at the end of this abstract. For the following description of the TSIMMIS architecture refer to Figure 1.

1.1 Translators and Common Model

Figure 1 shows a collection of (disk-shaped) heterogeneous information sources. Above each source is a *translator* (or *wrapper*) that logically converts the underlying data objects to a common information model. To do this logical translation, the translator converts

*This work was supported by ARPA Contract F33615-93-1-1339, by the Anderson Faculty Scholar Fund, by the Center for Integrated Systems at Stanford University, and by equipment grants from Digital Equipment Corporation and IBM Corporation. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of the US Government.

[1] As an acronym, TSIMMIS stands for "The Stanford-IBM Manager of Multiple Information Sources." In addition, Tsimmis is a Yiddish word for a stew with "heterogeneous" fruits and vegetables integrated into a surprisingly tasty whole.

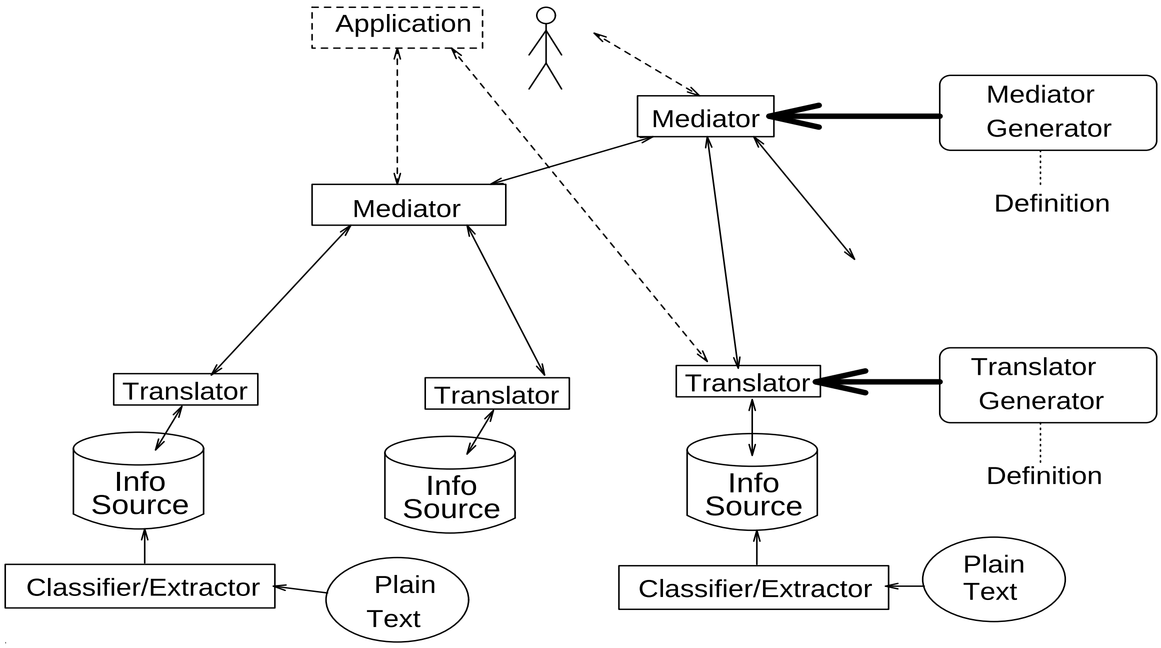

Figure 1: Tsimmis Architecture

queries over information in the common model into requests that the source can execute, and it converts the data returned by the source into the common model.

For the TSIMMIS project we have adopted a simple *self-describing* (or *tagged*) object model in which objects have labels, types, values, and an (optional) identifier. Similar models have been in use for years; we call our version the *Object Exchange Model*, or *OEM*. OEM allows simple nesting of objects; a complete specification is given in [10]. All objects and their subobjects have *labels* that describe their meaning. For example, the following object represents a Fahrenheit temperature of 80 degrees:

$\langle$temp-in-Fahrenheit, int, 80$\rangle$

where the string "temp-in-Fahrenheit" is a human-readable label, the type "int" indicates an integer value, and "80" is the value itself. If we wish to represent a complex object, then each component of the object has its own label. For example, an object representing a set of two temperatures may look like:

$\langle$set-of-temps, set, $\{cmp_1,\ cmp_2\}\rangle$
cmp_1: $\langle$temp-in-Fahrenheit, int, 80$\rangle$
cmp_2: $\langle$temp-in-Celsius, int, 20$\rangle$

OEM is very simple, while providing the expressive power and flexibility needed for integrating information from disparate sources. We also have developed a query language, *OEM-QL*, for requesting OEM objects. OEM-QL adapts existing SQL-like languages for object-oriented models (e.g., [6,7,8,11]) to OEM. Details and examples can be found in [10].

Note that many important information sources are completely unstructured, consisting of plain files or incoming bit strings (e.g., from a newswire). The *Classifier/Extractor* shown at the bottom of Figure 1 automatically classifies unstructured information and extracts key attributes. For example, the Classifier/Extractor might determine that a file is an e-mail message and extract the sender, receiver, date, and contents. The information collected by the Classifier/Extractor can then be exported via a translator to the rest of the TSIMMIS system, together with the raw data. The Classifier/Extractor component is based on the *Rufus* system developed at the IBM Almaden Research Center [12].

1.2 Mediators

Above the translators in Figure 1 lie the *mediators*. A mediator is a software module that refines in some way information from one or more sources [15]. A mediator embodies the knowledge that is necessary for processing a specific type of information. For example, a mediator for "current events" might know that relevant information sources are the AP Newswire and the New York Times database. When the mediator receives a query, say for articles on "Bosnia," it will know to forward the query to those sources. The mediator may also process answers before forwarding them to the user, say by converting dates to a common format, or by eliminating articles that duplicate information. While the task of converting dates is probably straightforward, the task of eliminating duplicate information could be very complex—figuring out that two articles written by different authors say "the same thing" requires real intelligence. In TSIMMIS we are focusing on relatively simple mediators based on patterns or rules. Still, even simple mediators can perform very useful information processing and merging tasks.

Implementing a mediator can be complicated and time-consuming, but we believe that much of the coding involved in mediators can be automated. Hence, one important goal of the TSIMMIS project is to generate mediators automatically or semi-automatically from high level descriptions of the information processing they need to do. This capability is represented by the *mediator generator* box on the right side of Figure 1. Similarly, we provide a *translator generator* that can generate OEM translators from on a description of the conversions that need to take place for queries received and results returned. This component, also illustrated in Figure 1, significantly facilitates the task of implementing a new translator.

1.3 System and User Interfaces

Mediators export an interface to their clients that is identical to that of translators. Both translators and mediators take as input OEM-QL queries and return OEM objects. Hence, end users and mediators can obtain their information from translators and/or other mediators. This approach allows new sources to become useful as soon as a translator is supplied. It allows mediators to access new sources transparently, and it allows mediators to be "stacked," performing more and more processing and refinement of the relevant information.

End users (top of Figure 1) can access information either by writing applications that request OEM objects, or by using one of the generic browsing tools we have developed. Our most recent browsing tool, called *MOBIE* (MOsaic Based Information Explorer) is a graphical user interface that provides access through *Mosaic* or other *World Wide Web* viewers [3, 14]. MOBIE lets end users connect to mediators or translators and specify queries using OEM-QL. Specifically, the user writes a query as an interactive world wide web page, or selects a query from a menu. The answer is received as a hypertext document. The root of this document shows one or more levels of the answer object, with hypertext links available to take the user to portions of the answer that did not appear on the root document. The goal of this object browsing component of TSIMMIS is to provide a platform-independent tool for displaying and exploring the OEM objects that are returned as a result of OEM-QL queries. Due to the nested structure of OEM objects, it is necessary to provide mechanisms that let end users navigate easily through the answer space, much as they would navigate through a tree structure.

An important advantage of using Mosaic as the basis for our user interface is its widespread use and popularity. (Mosaic currently operates on Unix workstations, on Macintosh computers, and on many PC's.) Hence, ultimately anyone on the internet should be able to use TSIMMIS and MOBIE to explore any information source on the net, provided there is an appropriate translator or mediator available for it.

2 Approach to Information Gathering in TSIMMIS

It is important to note that in TSIMMIS there is not a single global database containing the integrated information, nor even a global database schema. Furthermore, mediators and translators are not required to produce objects with a fixed schema or type, but each query determines a schema for objects that match the query. The schema-less approach in TSIMMIS is well suited for retrieving information from sources whose contents may change rapidly and without prior notice.

One of the key points of our approach to information gathering is the fact that no person or software component needs to have a global view of all the information handled by the system. For instance, to build a mediator, it is only necessary to understand the sources that the mediator will use (the same is obviously true for translators). Similarly, by combining mediators and stacking them into hierarchies, users or application programs can access all the information that is made available by each mediator without ever having to know where the relevant information is coming from and how to access it.

There are a number of differences between integration of information sources in the TSIMMIS project and other database integration efforts (e.g. [1, 2, 4, 5, 9, 13] and many others):

- TSIMMIS focuses on providing integrated access to very diverse and dynamic information. The information may be unstructured or semi-structured, often having no regular schema to describe it. The components of objects may vary in unpredictable ways (e.g., some pictures may be color, others black and white, others missing, some with captions and some without). Furthermore, the available sources, their contents, and the meaning of their contents may change frequently.

- Integration in our environment requires more human participation. In the extreme case, integration is performed manually by the end user. For example, a stock broker may read a report saying that IBM has named a new CEO, then retrieve recent IBM stock prices from a database to deduce that stock prices will rise. In other cases, integration may be automated by a mediator, but only after a human studies samples of the data, determines the procedure to follow, and develops an appropriate specification for the mediator generator.

- TSIMMIS assumes that information access and integration are intertwined. In a traditional integration scenario, there are two phases: an integration phase where data models and schemas (or parts thereof) are merged, and an access phase where data is fetched. In our environment, it may not be clear how information is merged until samples are viewed, and the integration strategy may change if certain unexpected data is encountered.

In summary, the TSIMMIS goal is *not* to perform fully automated information integration that hides all diversity from the user, but rather to provide a framework and tools to assist humans (end users and/or humans programming integration software) in their information processing and integration activities.

References

[1] R. Ahmed et al. The Pegasus heterogeneous multidatabase system. *IEEE Computer*, 24:19–27, 1991.

[2] Y. Arens, C. Y. Chee, C. Hsu, and C. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent & Cooperative Information Systems*, 2(2):127–158, June 1993. M. Papazoglou and T. Sellis, editors-in-chief.

[3] T. J. Berners-Lee, R. Cailliau, and J.F. Groff. The World Wide Web. *Computer Networks and ISDN Systems*, 25:454–459, 1992.

[4] A. Gupta. *Integration of Information Systems: Bridging Heterogeneous Databases.* IEEE Press, 1989.

[5] J. Hammer and D. McLeod. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *International Journal of Intelligent & Cooperative Information Systems*, 2(1):51–83, March 1993. M. Papazoglou and T. Sellis, editors-in-chief.

[6] M. Kifer, W. Kim, and Y. Sagiv. Querying object-oriented databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59–68, San Diego, California, June 1992.

[7] W. Kim et al. On resolving schematic heterogeneity in multidatabase systems. *Distributed And Parallel Databases*, 1:251–279, 1993.

[8] H. F. Korth and M. A. Roth. Query languages for nested relational databases. In *Nested Relations and Complex Objects in Databases*, pages 190–204. Springer-Verlag, 1989.

[9] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[10] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings Data Engineering Conference*, Taipei, Taiwan, March 1995.

[11] M. A. Roth, H. F. Korth, and A. Silberschatz. Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, 13:389–417, 1988.

[12] K. Shoens, A. Luniewski, P. Schwarz, J. Stamos, and J. Thomas. The RUFUS system: Information organization for semi-structured data. In *Proceedings of the International Conference on Very Large Databases*, pages 97–107, Dublin, Ireland, August 1993.

[13] G. Thomas et al. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 22:237–266, 1990.

[14] Steven J. Vaughan-Nichols. How to glue together Mosaic (internet browser) (tutorial). *Government Computer News*, 13(15):33, July 1994.

[15] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25:38–49, 1992.