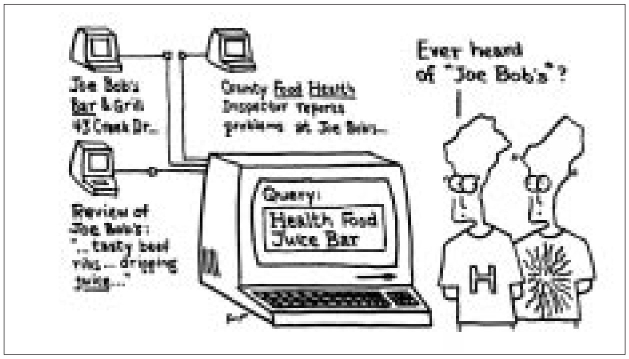

Joe Bob's Bar & Grill 43 Creek Dr...
County Food Health Inspector reports problems at Joe Bob's...
Review of Joe Bob's: "...tasty beef ribs... dripping juice..."
Query:
Health Food Juice Bar
Ever heard of "Joe Bob's"?
H

The Ariadne approach to Web-based information integration

Craig A. Knoblock and Steven Minton, University of Southern California

The rise of hyperlinked networks has made a wealth of data readily available. However, the Web's browsing paradigm does not strongly support retrieving and integrating data from multiple sites. Today, the only way to integrate the huge amount of available data is to build specialized applications, which are time-consuming, costly to build, and difficult to maintain. Mediator technology offers a solution to this dilemma. Information mediators,[1–4] such as the SIMS system,[5] provide an intermediate layer between information sources and users. Queries to a mediator are in a uniform language, independent of such factors as the distribution of information over sources, the source query languages, and the location of sources. The mediator determines which data sources to use, how to obtain the desired information, how and where to temporarily store and manipulate data, and how to efficiently retrieve information from the sources.

One of the most important ideas underlying information mediation in many systems, including SIMS, is that for each application there is a unifying *domain model* that provides a single ontology for the application. The domain model ties together the individual *source models*, which each describe the contents of a single information source. Given a query in terms of the domain model, the system dynamically selects an appropriate set of sources and then generates a plan to efficiently produce the requested data.

Information mediators were originally developed for integrating information in databases. Applying the mediator framework to the Web environment solves the difficult problem of gaining access to real-world data sources. The Web provides the underlying communication layer that makes it easy to set up a mediator system, because it is typically much easier to get access to Web data sources than to the underlying databases systems. In addition, the Web environment means that users who want to build their own mediator application need no expertise in installing, maintaining, and accessing databases.

We have developed a Web-based version of the SIMS mediator architecture, called Ariadne.[6] In Greek mythology, Ariadne was the daughter of Minos and Pasiphae who gave Theseus the thread with which to find his way out of the Minotaur's labyrinth. The Ariadne project's goal is to make it simple for users to create their own specialized Web-based mediators. We are developing the technology for rapidly constructing mediators to extract, query, and integrate data from Web sources. The system includes tools for constructing wrappers that make it possible to query Web sources as if they were databases and the mediator technology required to dynamically and efficiently answer queries using these sources.

A simple example illustrates how Ariadne can be used to provide access to Web-based sources (also see the "Ariadne" sidebar). Numerous sites provide reviews on restaurants, such as Zagats, Fodors, and CuisineNet, but none are comprehensive, and checking each site can be time consuming. In addition, information from other Web sources can be useful in selecting a restaurant. For example, the LA County Health Department publishes the health rating of all restaurants in the county, and many sources provide maps showing the location of restaurants. Using Ariadne, we can integrate these sources relatively easily to create an application where people could search these sources to create a map showing the restaurants that meet their requirements.

With such an application, a user could pose requests that would generate a map listing all the seafood restaurants in Santa Monica that have an "A" health rating and whose typical meal costs less than $30. The resulting map would let the user click on the individual restaurants to see the restaurant critic reviews. (In practice, we do not support natural language, so queries are either expressed in a structured query language or are entered through a Web-based graphical user interface.) The integration process that Ariadne facilitates can be complex. For example, to actually place a restaurant on a map requires the restaurant's latitude and longitude, which is not usually listed in a review site, but can be determined by running an online geocoder, such as Etak, which takes a street address and returns the coordinates.

Figure 3 outlines our general framework. We assume that a user building an application has identified a set of semistructured Web sources he or she wants to integrate. These might be both publicly available sources as well as a user's personal sources. For each source, the developer uses Ariadne to generate a wrapper for extracting information from that source. The source is then linked into a global, unified domain model. Once the mediator is constructed, users can query the mediator as if the sources were all in a single database. Ariadne will efficiently retrieve the requested information, hiding the planning and retrieval process details from the user.

Research challenges in Web-based integration

Web sources differ from databases in many significant ways, so we could not simply apply the existing SIMS system to integrate Web-based sources. Here we'll describe the problems that arise in the Web environment and how we addressed these problems in Ariadne.

Converting semistructured data into structured data. Web sources are not databases, but to integrate sources we must be able to query the sources as if they were. This is done using a *wrapper*, which is a piece of software that interprets a request (expressed in SQL or some other structured language) against a Web source and returns a structured reply (such as a set of tuples). Wrappers let the mediator both locate the Web pages that contain the desired information and extract the specific data off a page. The huge number of evolving Web sources makes manual construction of wrappers expensive, so we need the tools for rapidly building and maintaining wrappers.

For this, we have developed the Stalker inductive-learning system,[7] which learns a set of extraction rules for pulling information off a page. The user trains the system by marking up example pages to show the system what information it should extract

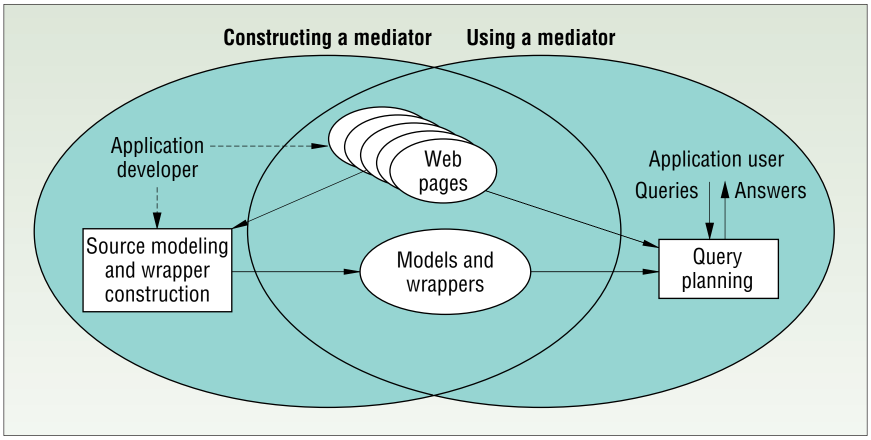


Figure 3. Architecture for information integration on the Web.

from each page. Stalker can learn rules from a relatively small number of examples by exploiting the fact that there are typically "landmarks" on a page that help users visually locate information.

Consider our restaurant mediator example. To extract data from the Zagats restaurant review site, a user would need to build two wrappers. The first lets the system extract the information from an index page, which lists all of the restaurants and contains the URLs to the restaurant review pages. The second wrapper extracts the detailed data about the restaurant, including the address, phone number, review, rating, and price. With these wrappers, the mediator can answer queries to Zagats, such as "find the price and review of Spago" or "give me the list of all restaurants that are reviewed in Zagats."

In his companion essay on the Information Manifold, Alon Levy claims that the problem of wrapping semistructured sources will soon be irrelevant because XML will eliminate the need for wrapper construction tools. We believe that he is being overly optimistic about the degree that XML will solve the wrapping problem. XML clearly is coming; it will significantly simplify the problem and might even eliminate the need for building wrappers for many Web sources. However, the problem of querying semistructured data will not disappear, for several reasons:

- There will always be applications where the providers of the data do not want to actively share their data with anyone who can access their Web page.
- Just as there are legacy Cobol programs, there will be legacy Web applications for many years to come.
- Within individual domains, XML will greatly simplify the access to sources; however, across domains people are unlikely to agree on the granularity that information should be modeled. For example, for many applications, the mailing address is the right level of granularity to model address, but if you want to geocode an address, it needs to be divided into street address, city, state, and zip code.

Planning to integrate data in the Web environment. Another problem that arises in the web environment is that generating efficient plans for processing data is difficult. For one, the number of sources to be integrated could be much larger than in the database environment. Also, Web sources do not provide the same processing capabilities found in a typical database system, such as the ability to perform joins. Finally, unlike relational databases, there might be restrictions on how a source can be accessed, such as a geocoder that takes the street address returns the geographic coordinates, but cannot take the geographic coordinates and return the street address.

Ariadne breaks down query processing into a preprocessing phase and a query-planning phase. In the first phase, the system determines the possible ways of combining the available sources to answer a query. Because sources might be overlapping—an attribute may be available from several sources—or replicated, the system must determine an appropriate combination of sources that can answer the query. The Ariadne source-selection algorithm[8] preprocesses the domain model so that the system can efficiently and dynamically select sources based on the classes and attributes mentioned in the query.

In the second phase, Ariadne generates a plan using a method called Planning-by-Rewriting.[9,10] This approach takes an initial, suboptimal plan and attempts to improve it by applying rewriting rules. With query planning, producing an initial, suboptimal plan is straightforward—the difficult part is finding an efficient plan. The rewriting process iteratively improves the initial query plan using a local search process that can change both the sources used to answer a query and the order of the operations on the data.

In our restaurant selection example, to answer queries that cover all restaurants, the system would need to integrate data from multiple sources (wrappers) for each restaurant review site and filter the resulting restaurant data based on the search parameters. The mediator would then geocode the addresses to place the data on a map. The plans for performing these operations might involve many steps, with many possible orderings and opportunities to exploit parallelism, in minimizing the overall time to obtain the data. Our planning approach provides a tractable approach to producing large, high-quality information-integration plans.

Providing fast access to slow Web sources. In exploiting and integrating Web-based information sources, accessing and extracting data from distributed Web sources is also much slower than retrieving information from local databases. Because the amount of data might be huge and the remote sources are frequently being updated, simply warehousing all of the data is not usually a practical option. Instead, we are working on an approach to selectively materialize (store locally) critical pieces of data that let the mediator efficiently perform the integration task. The materialized data might be portions of the data from an individual source or the result of integrating data from multiple sources.

To decide what information to store locally, we take several factors into account. First, we consider the queries that have been run against a mediator application. This lets the system focus on the portions of the data that will have the greatest impact on the most queries. Next, we consider both the frequency of updates to the sources and the application's requirements for getting the most recent information. For example, in the restaurant application, even though reviews might change daily, providing information that is current within a week is probably satisfactory. But, in a finance application, providing the latest stock price would likely be critical. Finally, we consider the sources' organization and structure. For example, the system can only get the latitude and longitude from the geocoder by providing the street address. If the application lets a user request the restaurants located within a region of a map, it could be very expensive to figure out which restaurants are in that region because the system would need to geocode each restaurant to determine whether it falls within the region. Materializing the restaurant addresses and their corresponding geocodes avoids a costly lookup.

Once the system decides to materialize a set of information, the materialized data becomes another information source for the mediator. This meshes well with our mediator framework because the planner dynamically selects the sources and the plans that can most efficiently produce the requested data. In the restaurant example, if the system decides to materialize address and geocode, it can use the locally stored data to determine which restaurants could possibly fall within a region for a map-based query.

Resolving naming inconsistencies across sources. Within a single site, entities—such as people, places, countries, or companies—are usually named consistently. However, across sites, the same entities might be referred to with different names. For example, one restaurant review site might refer to a restaurant as Art's Deli and another site might call it Art's Delicatessen. Or, one site might use California Pizza Kitchen and another site could use the abbreviation CPK. To make sense of data that spans multiple sites, our system must be able to recognize and resolve these differences.

In our approach, we select a primary source for an entity's name and then provide a mapping from that source to each of the other sources that use a different naming scheme. The Ariadne architecture lets us represent the mapping itself as simply another wrapped information source. Specifically, we can create a *mapping table*, which specifies for each entry in one data source what the equivalent entity is called in another data source. Alternatively, if the mapping is computable, Ariadne can represent the mapping by a *mapping function*, which is a program that converts one form into another form.

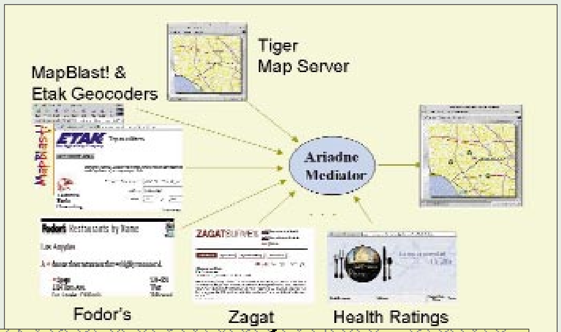
Ariadne

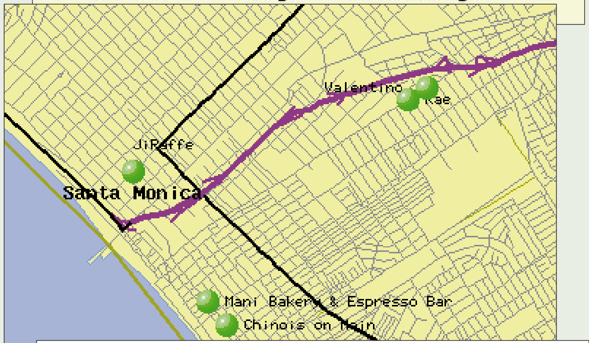
This Restaurant Location application of Ariadne shown in the first image integrates data from a variety of sources, including restaurant review sites, health ratings, geocoders, and maps.

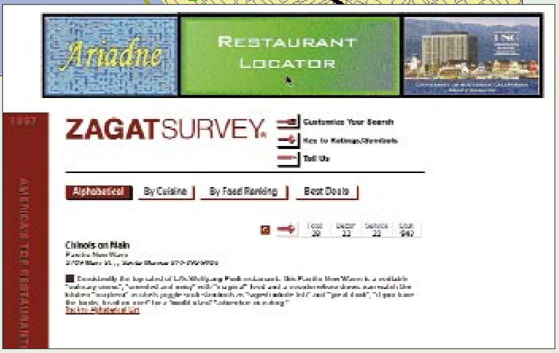
In response to a query for all highly rated restaurants in Santa Monica with an 'A' health rating, the mediator finds the restaurants that satisfy the query by extracting the data directly from the relevant Web sites.

The mediator also produces a map of the restaurants (second image) by converting the street addresses into latitude and longitude coordinates using an online geocoder.

Each point on the map in the second image is clickable. Selecting the point for Chinois on Main returns the detailed restaurant review directly from the appropriate restaurant review site (third image).







We are developing a semi-automated method for building mapping tables and functions by analyzing the underlying data in advance. The basic idea is to use information-retrieval techniques, such as those described in William Cohen's companion essay, to provide an initial mapping,[11] and then use additional data in the sources to resolve any remaining ambiguities via statistical learning methods.[12] For example, restaurants are best matched up by considering name, street address, and phone number, but not by using a field such as city because a restaurant in Hollywood could be listed as either being in Hollywood or Los Angeles and different sites list them differently.

The future of Web-based integration

As more and more data becomes available, users will become increasingly less satisfied using existing search engines that return massive quantities of mostly irrelevant information. Instead, the Web will move toward more specialized content-based applications that do more than simply return documents. Information-integration systems such as Ariadne will help users rapidly construct and extend their own Web-based applications out of the huge quantity of data available online.

While information integration has made tremendous progress over the last few years,[13] many hard problems still must be solved. In particular, two mostly overlooked problems deserve more attention:

- Coming up with the models or source descriptions of the information sources, a time-consuming and difficult problem that is largely performed by hand today.
- Automatically locating and integrating new sources of data, which would be enabled by solutions to the first problem. (This problem has been addressed in limited domains, such as Internet shopping,[14] but the problem is still largely unexplored.)

For more information on the Ariadne

project and example applications that were built using Ariadne, see the Ariadne homepage at *http://www.isi.edu/ariadne*.

References

1. G. Wiederhold, “Mediators in the Architecture of Future Information Systems,” *Computer*, Vol. 25, No. 3, Mar. 1992, pp. 38–49.
2. H. Garcia-Molina et al., “The Tsimmis Approach to Mediation: Data Models and Languages,” *J. Intelligent Information Systems*, 1997.
3. A.Y. Levy, A. Rajaraman, and J.J. Ordille, “Querying Heterogeneous Information Sources Using Source Descriptions,” *Proc. 22nd Very Large Databases Conf.*, Morgan Kaufmann, San Francisco, 1996, pp. 251–262.
4. M.R. Genesereth, A.M. Keller, and O.M. Duschka, “Infomaster: An Information Integration System,” *Proc. ACM Sigmod Int’l Conf. Management of Data*, ACM Press, New York, 1997, pp. 539–542.
5. Y. Arens, C.A. Knoblock, and W.-M. Shen, “Query Reformulation for Dynamic Information Integration,” *J. Intelligent Information Systems*, Special Issue on Intelligent Information Integration, Vol. 6, Nos. 1 and 3, 1996, pp. 99–130.
6. C.A. Knoblock et al., “Modeling Web Sources for Information Integration,” *Proc. 11th Nat’l Conf. Artificial Intelligence*, AAAI Press, Menlo Park, Calif., 1998, pp. 211–218.
7. I. Muslea, S. Minton, and C.A. Knoblock, “Stalker: Learning Extraciton Rules for Semistructured Web-Based Information Sources,” *Proc. 1998 Workshop AI and Information Integration*, AAAI Press, 1998, pp. 74–81.
8. J.L. Ambite et al., *Compiling Source Descriptions for Efficient and Flexible Information Integration*, tech. report, Information Sciences Institute, Univ. of Southern California, Marina del Rey, Calif., 1998.
9. J.L. Ambite and C.A. Knoblock, “Planning by Rewriting: Efficiently Generating High-Quality Plans,” *Proc. 14th Nat’l Conf. Artificial Intelligence*, AAAI Press, 1997, pp. 706–713.
10. J.L. Ambite and C.A. Knoblock, “Flexible and Scalable Query Planning in Distributed and Heterogeneous Environments,” *Proc. Fourth Int’l Conf. Artificial Intelligence Planning Systems*, AAAI Press, 1998, pp. 3–10.
11. W.W. Cohen, “Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity,” *Proc. ACM Sigmod-98*, ACM Press, 1998, pp. 201–212.
12. T. Huang and S. Russell, “Object Identification in a Bayesian Context,” *Proc. 15th Int’l J. Conf. AI*, Morgan Kaufmann, 1997, pp. 1276–1283.
13. *Proc. 1998 Workshop on AI and Information Integration*, AAAI Press, 1998.
14. R.B. Doorenbos, O. Etzioni, and D.S. Weld, “A Scalable Comparison-Shopping Agent for the World-Wide Web,” *Proc. First Int’l Conf. Autonomous Agents*, AAAI Press, 1997, pp. 39–48.