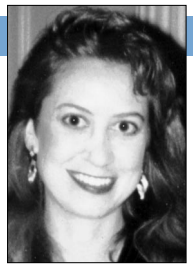


Information integration

By Marti A. Hearst
University of California, Berkeley
hearst@sims.berkeley.edu

Despite the Web's current disorganized and anarchic state, many AI researchers believe that it will become the world's largest knowledge base. In this installment of Trends and Controversies, we examine a line of research whose final goal is to make disparate data sources work together to better serve users' information needs. This work is known as *information integration*. In the following essays, our authors talk about its application to datasets made available over the Web.

Alon Levy leads off by discussing the relationship between information-integration and traditional database systems. He then enumerates important issues in the field and demonstrates how the Information Manifold project has addressed some of these, including a language for describing the contents of diverse sources and optimizing queries across sources.

Craig Knoblock and Steve Minton describe the Ariadne system. Two of its distinguishing features are its use of *wrapper* algorithms to extract structured information from semistructured data sources and its use of planning algorithms to determine how to integrate information efficiently and effectively across sources. This system also features a mechanism that determines when to prefetch data depending on how often the target sources are updated and how fast the databases are.

William Cohen describes an interesting variation on the theme, focusing on "informal" information integration. The idea is that, as in related fields that deal with uncertain and incomplete information, an information-integration system should be allowed to take chances and make mistakes. His Whirl system uses information-retrieval algorithms to find approximate matches between different databases, and as a consequence knits together data from quite diverse sources.

A controversy emerges in the midst of this trend, centering around the issue of whether information extraction from HTML-based Web pages is a long-standing problem. Proponents of XML (Extensible Markup Language, see *www.w3.org/TR/REC-xml.html*) argue that in the future information of any importance will be exchanged between programs using a well-defined protocol, rather than being displayed solely for purposes of reading using ad hoc formats in HTML. In his essay, Levy argues that the problem of extracting information from HTML markup will, as a consequence of such protocols, become less important. He notes, however, that the problem of integrating data that differs semantically will still remain. Knoblock and Minton counter that the need for HTML wrappers will remain strong, arguing that there will always be exceptions and legacy pages.

Cohen takes a different stance, suggesting that many information providers want to help inform people, but might not see a direct benefit from the investment required to form a highly structured data source. He suggests that cheap, approximate information integration, such as enabled by his system, can render these simpler sites more powerful, providing a larger benefit than any individual site developer alone could attain, and getting around the chicken-and-egg problem of who pays to make useful information available free.

On a different note, Haym Hirsh of Rutgers has signed on to help edit Trends & Controversies. To continue providing sharp, cogent debates on topics that span a wide range of intelligent systems research and applications development, he and I will be alternating installments. For his first outing next issue, Haym has lined up Barbara Hayes-Roth, Janet Murray, and Andrew Stern, who will address interactive fiction.

—Marti Hearst

The Information Manifold approach to data integration

Alon Y. Levy, University of Washington

A data-integration system provides a uniform interface to a multitude of data sources. Consider a data-integration system providing information about movies from data sources on the World Wide Web. There are numerous sources on the Web concerning movies, such as the Internet Movie Database (which provides comprehensive listings of movies, their casts, directors, genres, and so forth), MovieLink (listing playing times of movies in US cities), and several sites that provide textual reviews for selected movies. Suppose we want to find which Woody Allen movies are playing tonight in Seattle and see their respective reviews. None of these data sources in isolation can answer this query. However, by combining data from multiple sources, we can answer queries like this one, and even more complex ones.

To answer our query, we would first query the Internet Movie Database to obtain the list of movies directed by Woody Allen, and then feed the result into the MovieLink database to check which ones are playing in Seattle. Finally, we would find reviews for the relevant movies using any of the movie review sites.

Most importantly, a data-integration system lets users focus on specifying what they want, rather than thinking about how to obtain the answers. As a result, it frees them from the tedious tasks of finding the relevant data sources, interacting with each source in isolation using a particular interface, and combining data from multiple sources.

Traditional database systems

To understand the challenges involved in building data-integration systems, I will briefly compare the problems that arise in this context with those encountered in traditional database systems. In this discussion, I focus mainly on comparisons with relational database systems, but the differences also hold for systems based on other models, such as object-oriented and object-relational ones. Figure 1 illustrates the different stages in processing a query in a data-integration system.

Data modeling. Traditional database systems and data-integration systems differ mainly in the process they use to organize data into an application. In a traditional system, the application designer examines the application's requirements, designs a database schema (such as a set of relation names and the attributes of each relation), and then implements the application, part of which involves actually populating the database (inserting tuples into the tables).

In contrast, a data-integration application begins from a set of pre-existing data sources. These sources might be database systems, but more often are unconventional data sources, such as structured files, legacy

systems, or Web sites. Here the application builder must design a *mediated schema* on which users will pose queries. The mediated schema is a set of virtual relations, in that they are not actually stored anywhere. The mediated schema is designed manually for a particular data-integration application. For example, in the movie domain, the mediated schema might contain the relations `MovieInfo(id, title, genre, country, year, director)` describing the different properties of a movie, the relation `MovieActor(id, name)` representing a movie's cast, and `MovieReview(id, review)` representing reviews of movies.

Along with the mediated schema, the application designer needs to supply descriptions of the data sources. The descriptions specify the relationship between the relations in the mediated schema and those in the local schemas at the sources. (Even though not all the sources are databases, we model them as having schemas at the conceptual level.) An information-source description specifies

- the source's contents (for example, contains movies),
- the attributes found in the source (genre, cast),
- constraints on the source's contents (contains only American movies),
- the source's completeness and reliability, and finally,
- its query-processing capabilities (can perform selections or can answer arbitrary SQL queries).

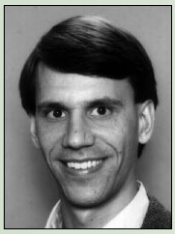
Because the data sources are preexisting, data in the sources might be overlapping and even contradictory. Furthermore, we might face the following problems:

- *Semantic mismatches between sources.* Because each data source has been designed by a different organization for different purposes, the data is modeled in different ways. For example, one source might store a relational database that stores all of a particular movie's attributes in one table, while another source might spread the attributes across several relations. Furthermore, the names of the attributes and tables will differ from one source to another, as will the choice of what should be a table and what should be an attribute.
- *Different naming conventions.* Sources use different names to refer to the same object. For example, the same person might be called as "John Smith" in one source and "J.M. Smith" in another.

Query optimization and execution. A traditional relational-database system accepts a *declarative* query in SQL. The system first parses the query before passing it to the *query optimizer*. The optimizer produces an efficient *query-execution plan* for the query, which is an imperative program that specifies exactly how to evaluate the query. In particular, the plan specifies the order for performing the query's operations (join, selection, and projection), the method for implementing each operation (such as sort-merge join or hash join), and the scheduling of the different operators (where parallelism is possible). Typically, the optimizer selects a query-execution plan by searching a space of possible plans and comparing their estimated costs. To evaluate a query-execution plan's cost, the optimizer relies on extensive statistics about the underlying data, such as the sizes of relations, sizes of domains, and selectivity of predicates. Finally, the query-execution plan passes to the query-execution engine, which evaluates the query.

The traditional database and the data-integration contexts differ primarily in that the optimizer has little information about the data, because the data resides in remote autonomous sources rather than locally. Furthermore, because the data sources are not necessarily database systems, the sources appear to have different processing capabilities. For example, one data source might be a Web interface to a legacy information system, while another might be a program that scans data stored in a structured file (such as bibliography entries). Hence, the query optimizer must consider the possibility of exploiting a data source's query-processing capabilities. Query optimizers in distributed database systems also consider where parts of the query are executed, but in that context

Alon Y. Levy is a faculty member at the University of Washington's Computer Science and Engineering Department. His research interests are Web-site management, data integration, query optimization, management of semistructured data, description logics and their relationship to database query languages, abstractions and approximations of computational theories, and relevance reasoning. He received his PhD in computer science from Stanford University and his undergraduate degree at Hebrew University. Contact him at the Dept. of Computer Science and Engineering, Sieg Hall, Room 310, Univ. of Washington, Seattle, WA, 98195; alon@cs.washington.edu; http://www.cs.washington.edu/homes/alon/.



Craig Knoblock is a project leader and senior research scientist at the Information Sciences Institute, a research assistant professor in the Computer Science Department, and a key investigator in the Integrated Media Systems Center at the University of Southern California. His research interests include information gathering and integration, automated planning, machine learning, knowledge discovery, and knowledge representation. He received his BS from Syracuse University and his MS and PhD from Carnegie Mellon University, all in computer science. Contact him at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292; knoblock@isi.edu; http://www.isi.edu/~knoblock.

Steve Minton is a senior computer scientist at the Information Sciences Institute and a research associate professor in the Computer Science Department at the University of Southern California. His research interests are in machine learning, planning, scheduling, constraint-based reasoning, and program synthesis. He received his BA in psychology from Yale University and his PhD in computer science from Carnegie Mellon University. He founded the *Journal of Artificial Intelligence Research* and served as its first executive editor. He was recently elected to be a fellow of the AAAI. Contact him at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292; minton@isi.edu; http://www.isi.edu/sims/minton/homepage.html.

William Cohen is a principal research staff member in the department of Machine Learning and Information Retrieval Research at AT&T Labs-Research. In addition to information integration, his research interests include machine learning, text categorization, learning from large datasets, computational learning theory, and inductive logic programming. He received a bachelor's degree from Duke and a PhD from Rutgers, both in computer science. Contact him at AT&T Labs-Research, 180 Park Avenue, Florham Park NJ 07932-0971; wcohen@research.att.com; http://www.research.att.com/~wcohen/.

the different processors have identical capabilities. Finally, because data must be transferred over a network, the query optimizer and the execution engine must be able to adapt to data-transfer delays.

Query reformulation. A data-integration system user poses queries in terms of the mediated schema, rather than directly in the schema where the data resides. As a consequence, a data-integration system must first reformulate a user query into a query that refers directly to the schemas in the sources. Such a reformulation step does not exist in traditional database systems. To perform the reformulation step, the data-integration system uses the source descriptions.

Wrappers. Unlike a traditional query-execution engine that communicates with the storage manager to fetch the data, a data-integration system's query-execution plan must obtain data from remote sources. To do so, the execution engine communicates with a set of *wrappers*. A wrapper is a program that is specific to every data source and that translates the source's data to a form that the system's query processor can further process. For example, the wrapper might extract a set of tuples from an HTML file and perform translations in the data's format.

Semistructured data. The term *semistructured data* has been used with various meanings to refer to characteristics of data present in a data-integration system. To understand the importance of semistructured data, we distinguish between a lack of structure at the physical level versus one at the logical level. With lack of structure at the physical level, structured data (for example, tuples) are embedded in a file containing additional markup information such as HTML files. Extracting the actual values from the HTML file can be very complex task, and is one that the source's wrapper performs.

Most work on semistructured data concerns lack of structure at the logical level. In this context, semistructured data refers to cases in which the data does not necessarily fit into a rigidly predefined schema, as required in traditional database systems. This might arise because the data is very irregular and hence can be described only by a relatively large schema. In other cases, the schema might be rapidly evolving, or not even declared at all—it might be implicit in the data. The database community has developed several methods to model and query semistructured data and is currently considering the issues of query optimization and storage for such data.[1,2] Building data-integration systems based on a semistructured data model has two main advantages:

- In many cases, the data in the sources is indeed semistructured at the logical level.
- The models developed for semistructured data can cleanly integrate data coming from multiple data models, such as relational, object-oriented, and Web-data models.

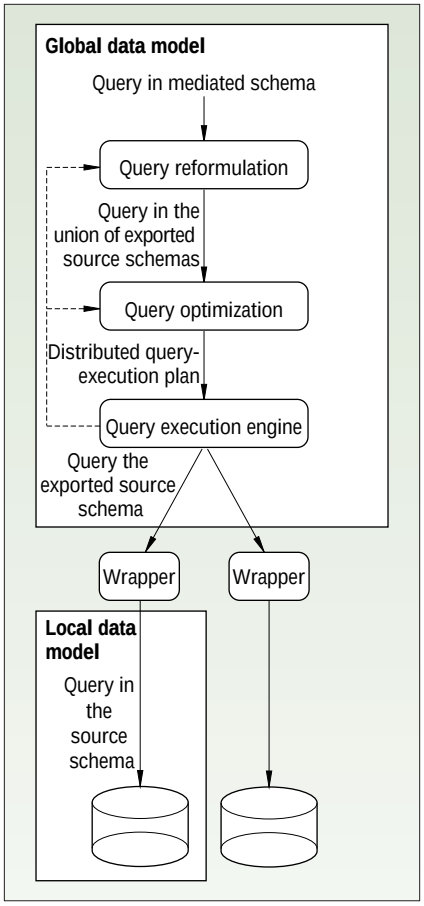


Figure 1. Prototypical architecture of a data-integration system.

Classes of data integration applications. The two main classes of data-integration applications are integration of data sources on the Web and within a single company or enterprise. In the latter case, the sources are not as autonomous as they are on the Web, but the requirements from a data-integration system might be more stringent.

The Information Manifold Project

In this project, we wanted to develop a system that would flexibly integrate many data sources with closely related content and that would answer queries efficiently by accessing only the sources relevant to the query. The remainder of this essay will describe its main contributions.[3–6]

The AI and DB approach. We based our approach in designing the Information Manifold on the observation that the data-integration problem lies at the intersection of database systems and artificial intelligence. Hence, we searched for solutions that combine and extend techniques from both fields. For example, we developed a representation language and a language for describing data sources that was simple from the knowledge-representation perspective, but that had the necessary added flexibility concerning previous techniques developed in the database community.

Source description language. The Information Manifold is most importantly a flexible mechanism for describing data sources. This mechanism lets users describe complex constraints on a data source's contents, thereby letting them distinguish between sources with closely related data. Also, this mechanism makes it easy to add or delete data sources from the system without changing the descriptions of other sources. Informally, the contents of a data source are described by a query over the mediated schema. For example, we might describe a data source as containing American movies that are all comedies and were produced after 1965 (source 1 in Figure 2). As another example, we can describe sources in whose schema significantly differs from the one in the mediated schema. For instance, we can describe a source in which a movie's `year`, `genre`, `actor`, and `review` attributes already appear in one table (source 2 in Figure 2). This source is modeled as containing the result of a join over a set of relations in the mediated schema. For some queries, extracting data from this source might be cheaper than from others if the join computed in the source is indeed required for the query.

The Information Manifold employed an expressive language, Carin, for formulating queries and for representing background knowledge about the relations in the mediated schema. Cairn[7] combined the expressive power of the datalog database-query language (needed to model relational sources) and Description Logics, which are knowledge-representation languages de-

signed especially to model complex hierarchies that frequently arise in data-integration applications.

Source 1
```
select title, year, director
from MovieInfo
where genre = comedy
      year ≥ 1965
      country = USA
```

Source 2
```
select title, genre, review
from MovieInfo m, MovieReview r
     where m.id = r.id
```

Figure 2. Data-source descriptions.

Query-answering algorithms. We developed algorithms for answering queries using the information sources. Recall that user queries are posed in terms of the mediated schema. Hence, the main challenge in designing the query-answering algorithms is to reformulate the query such that it refers to the relations in the data sources. Our algorithms were the first to guarantee that only the relevant set of data sources are accessed when answering a query, even in the presence of sources described by complex constraints.

It is interesting to note the difference between our approach to query answering and that employed in the SIMS and Ariadne projects described in Craig Knoblock's and Steve Minton's companion essay. In their approach, even though they used a knowledge-representation system for specifying the source descriptions, they used a general-purpose planner to reformulate a user query into a query on the data sources. In contrast, our approach uses the reasoning mechanisms associated with the underlying knowledge-representation system to perform the reformulation. Aside from the natural advantages obtained by treating the representation and the query reformulation within the same framework, our approach can provide better formal guarantees on the results and can benefit immediately from extensions to the underlying knowledge-representation system.

Handling completeness information. In general, sources on the Web are not necessarily complete for the domain they are covering. For example, a computer science bibliography source is unlikely to contain all the references in the field. However, in some cases, we can assert local completeness statements about sources.[6] For example, the DB&LP Database (*http://www.informatik.uni-trier.de/ley/db/*) contains the complete set of papers published in some of the major database conferences. Knowledge of a Web source's completeness can help a data-integration system in several ways. Most importantly, because a negative answer from a complete source is meaningful, the data-integration system can prune access to other sources. In the Information Manifold, we developed a method for representing local-source completeness and an algorithm for exploiting such information in query answering.[5]

Using probabilistic information. The Information Manifold pioneered the use of probabilistic reasoning for data integration (representing another example of the combined AI and DB approach to the data-integration problem).[6] When numerous data sources are relevant to a given query (such as bibliographic databases available for a topic search), a data-integration system needs to order the access to the data sources. Such an ordering is dependent on the overlap between the sources and the query and on the coverage of the sources. We developed a probabilistic formalism for specifying and deducing such information and algorithms for ordering the access to data sources given such information.[6]

Exploiting source capabilities. Sources often have different query-processing capabilities. For example, one source might be a full-fledged relational database, while another might be a Web site with a very specific form interface that supports only a limited set of queries and that requires certain inputs be provided to it. The Information Manifold developed several novel algorithms for adapting to differing source capabilities. When possible, to reduce the amount of processing done locally, the system would fully exploit the query-processing capabilities of its data sources.[4,9] In addition, we developed a mechanism for describing source capabilities that is a natural extension of our method for describing source contents. [9]

What we did as a community

In the past few years, each of the groups working on data integration has made significant individual progress (see this magazine's Web page for a list of projects; *http://computer.org/intelligent)*. However, we have also made progress as a community. In particular, we have developed a common set of terms and dimensions along which we can now compare our work more rigorously.[12] For example, we can now compare the relative expressive power of our source-description languages. We have a set of properties along which we can compare our query-answering algorithms (such as, do they guarantee accessing only relevant sources or a minimal number of sources?). We can also compare features of our systems (do they assume sources are complete, can they handle local completeness, and can they compare directly between sources?).

We need to take this progress into account as we address the challenges that lie ahead. Our common terminology will enable us (and should force us) to compare systems more rigorously, either theoretically or experimentally. To proceed, we must also develop a set of data-integration benchmarks, along which we can experimentally compare our data-integration systems.

The immediate future

The data-integration problem is by no means solved. We have made significant progress in the problems relating to modeling data sources and developing methods for combining data from them via a single, integrated view. Many problems remain in that area, the most significant being the problem of name matching across sources. This problem is finally starting to be addressed in a principled manner in the Whirl system.[13]

In the near future, I believe that the bulk of the work in the field should shift into other, less attended problems, some of which I describe here.

Information presentation. Users are rarely interested in data that is simply and concisely presented. More commonly, the result of users queries are best seen as entry points into entire webs of data. This observation begs the question of how to build systems that enable us to flexibly design a web of information. In fact, this is exactly the problem we face in designing a richly structured Web site. The key to designing such systems is a declarative representation of a web of information's structure. Based on such a representation, we can easily specify how to restructure the information integrated from multiple sources into a structure that users can navigate. Recently, we have developed the Strudel system,[14] which is the first to apply these principles in creating Web sites.

Optimization and execution. Modern database systems succeed largely based on the careful design of their query-optimization and query-execution engines. Recall that the query optimizer is the module in charge of transforming a declarative query (given, for example, in SQL) into a query-execution plan, thereby making decisions on the order of joins and the specific methods for implementing each operation in the plan. The query-execution engine actually evaluates the plan. Given that data integration is a more general form of the problem addressed in database systems, they will succeed only if we carefully consider the design of these components in data-integration systems.

Two factors complicate the problems of query optimization and execution in the context of data integration: lack of extensive statistics on the data we are accessing (unlike with relational databases) and unpredictable arrival rates of data from the sources at runtime. Here, too, a combination of techniques from AI and database systems is likely to provide interesting solutions. In particular, in this context, the need for interleaving query optimization and query execution is much more significant. The idea of interleaving of planning and execution has been considered in the AI planning literature in recent years.[15] In contrast, current database systems perform complete query optimization before beginning the execution. The issues of query optimization and execution are the focus of the Tukwila project underway at the University of Washington.

Obtaining source descriptions. Current systems are very good at using descriptions of the source for answering queries. However, source descriptions must still be given manually. Specifically, the problem is to obtain the semantic mapping between the content of the source and the relations in the mediated schema. If data-integration systems are really going to scale up to large numbers, we must develop automatic methods for obtaining source descriptions, possibly by employing techniques from machine learning.

A nonproblem. Many data-integration efforts have focused on the problem of extracting data from HTML pages—extracting tuples from documents in which data is semistructured at the physical level. This problem will become significantly less important, given the emergence of standards such XML and languages that will facilitate querying XML documents.[15] Web sites that serve significant amounts of data are usually developed using some tool for serving database contents. Using such tools will make it easier to serve the data in XML form, rather than directly in HTML. Hence, Web sites will be able to export data in XML with no added burden to the information providers. Of course, some Web sites that do not want their data to be used for integration purposes might still only serve HTML pages, but trying to integrate data from such sources is probably a futile effort at best.

However, while the availability of data in XML format will reduce the emphasis on wrappers converting human-readable data to machine-readable data, the challenges of semantic integration I've mentioned and the need to manage data that is structured at the logical level remains. Furthermore, the machine-learning algorithms developed for extracting data from HTML pages might prove useful for the problem of obtaining semantic mappings.

Farther down the road

Once we can build stand-alone, robust, data-integration systems, we will face the challenge of embedding such systems in more general environments. I illustrate this challenge with two examples. The first concerns extending the interaction with a data-integration system beyond simple query answering. In particular, we should be able to use the data to automate some of the tasks we routinely perform with the data. For example, the system should be able to use the data for everyday information-management tasks, such as managing our personal information (our schedules, for example) and document workflow in organizations, or for alerting different users on important events.

The second method concerns the expectations we have from the data-integration system. It is unlikely that we will be able to answer all user queries fully automatically, because there will always remain sources for which we will have only models or sources whose structure (such as natural-language text) does not enable us to reliably extract the data. Hence, we must develop an environment in which the system cooperates with the user to obtain the answer to the query. Where possible, the system will answer the query completely; in other cases, the system will guide the user to the desirable answers.

References

1. P. Buneman, "Semistructured Data," *Proc. ACM Sigact-Sigmod-Sigart Symp. Principles of Database Systems (PODS)*, ACM Press, New York, 1997, pp. 117–121.
2. S. Abiteboul, "Querying Semi-Structured Data," *Proc. Int'l Conf. on Database Theory (ICDT)*, 1997.
3. A.Y. Levy, A. Rajaraman, and J.J. Ordille, "Query Answering Algorithms for Information Agents," *Proc. 11th Nat'l Conf. AI*, AAAI Press, Menlo Park, Calif., 1996, pp. 40–47.
4. A.Y. Levy, A. Rajaraman, and J.J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions," *Proc. 22nd Int'l Conf. Very Large Databases (VLDB-96)*, Morgan Kaufmann, San Francisco, 1996, pp. 251–262.
5. A.Y. Levy, "Obtaining Complete Answers from Incomplete Databases," *Proc. 22nd Int'l Conf. Very Large Databases (VLDB-96)*, Morgan Kaufmann, 1996.
6. D. Florescu, D. Koller, and A.Y. Levy, "Using Probabilistic Information in Data Integration," *Proc. 23nd Int'l Conf. Very Large Databases*, Morgan Kaufmann, 1997, pp. 216–225.
7. A.Y. Levy and M.-C. Rousset, "CARIN: A Representation Language Integrating Rules and Description Logics," *Proc. Int'l Description Logics*, 1995.
8. O. Etzioni and D. Weld, "A Softbot-Based Interface to the Internet," *Comm. ACM*, Vol. 37, No. 7, 1994, pp. 72–76.
9. A.Y. Levy, A. Rajaraman, and J.D. Ullman, "Answering Queries Using Limited External Processors," *Proc. ACM Sigact-Sigmod-Sigart Symp. Principles of Database Systems (PODS)*, ACM Press, 1996.
10. H. Garcia-Molina et al., "The TSIMMIS Approach to Mediation: Data Models and Languages (Extended Abstract)," *Next Generation Information Technologies and Systems (NGITS-95)*, 1995.
11. O. Etzioni and D. Weld, "A Softbot-Based Interface to the Internet," *Comm. ACM*, Vol. 37, No. 7, 1994, pp. 72–76.
12. D. Florescu, A. Levy, and A. Mendelzon, "Database Techniques for the World-Wide Web: A Survey," *Proc. ACM Sigmod-98*, ACM Press, 1998.
13. 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.
14. M. Fernandez et al., "Catching the Boat with Strudel: Experiences with a Web-site Management System," *Proc. ACM Sigmod-98*, ACM Press, 1998.
15. J. Ambros-Ingerson and S. Steel, "Integrating Planning, Execution, and Monitoring," *Proc. 13th Nat'l Conf. AI*, AAAI Press, Menlo Park, Calif., 1998, pp. 83–88.

The Ariadne approach to Web-based information integration

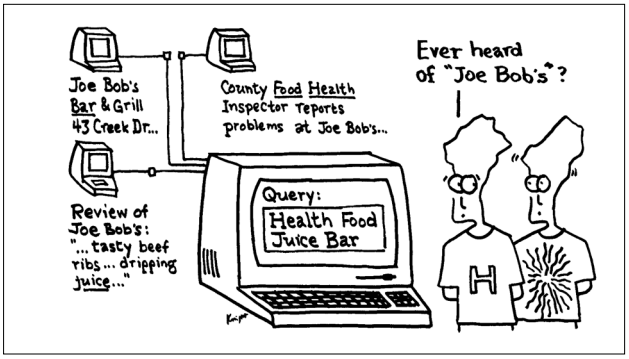
Craig A. Knoblock and Steven Minton, University of Southern California

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

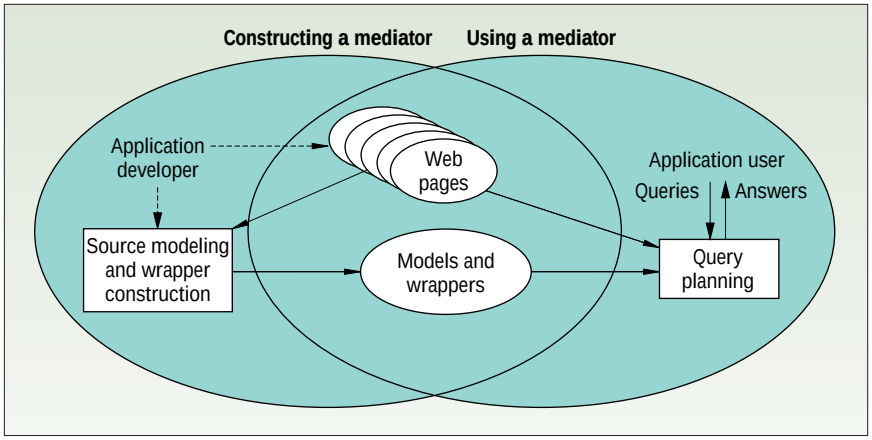


Figure 3. Architecture for information integration on the Web.

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.

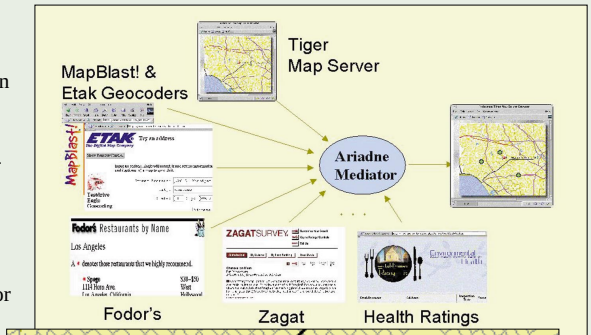
Ariadne

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 latitute 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.

The Whirl approach to information integration

William W. Cohen, AT&T Labs-Research

Search engines such as AltaVista and portal sites such as Yahoo! help us find useful online information sources. What we need now are systems to help use this information effectively. Ideally, we would like programs that answer a user's questions based on information obtained from many different online sources. We call such a program an information-integration system, because to answer questions it must integrate the information from the various sources into a single, coherent whole.

For example, consider consumer information about computer games. Many Web sites contain information of this sort. As this essay will show, in addition to the obvious benefit of reducing the number of sites a user must visit, integrating this information has several important and nonobvious advantages.

One advantage is that often, more questions can be answered using the integrated information than using any single source. Consider two sources containing slightly different information: one source categorizes games into children's games and adult games, and another categorizes games into arcade games, puzzle games, and adventure games. In this case, the sources must be integrated to find, say, a list of children's adventure games. Conversely, integration can help exploit overlap among sources; for instance, one might be interested in finding games that three or more sources have rated highly, or in reading several independent reviews of a particular game.

A second and more important advantage of integration is that making it possible to combine information sources also makes it possible to decompose information so as to represent it in a clean, modular way. For example, suppose we wished to create a Web site providing some new sort of information about computer games—say, information about which games work well on older, slower machines. The simplest way of representing this information is extensionally, as a list of games having this property. By itself, however, such a list is not very valuable to end users, who are probably interested in games that not only work on their PC, but also satisfy other properties, such as being inexpensive or well-designed. To make the list more useful, we are tempted to add additional structure—for instance, we might organize the games in the list into categories and provide, for each game, links to online resources, such as pricing information and reviews.

From the standpoint of computer science, augmenting the list of games in this way is clearly a bad idea, because it leads to a structure that lacks *modularity*. The original structure was a static, easily maintained list of computer games. In the augmented hypertext, this information is intermixed with orthogonal information about game categories, possibly ephemeral information concerning the organization of external Web sites, and possibly incorrect assumptions about the readers' goals. The resulting structure is hard to maintain and hard to modify in certain natural ways, such as by changing the set of categories used to organize the list of games.

To summarize, the simple, modular encoding of this information will be difficult for users to exploit, and the easy-to-use encoding will be difficult to create, modify, and maintain. By contrast, it is trivial to encode this information in a relational database in a manner that is both modular and useful: we simply create a relation listing all old-PC-friendly games, and standard query languages let users find, say, reviews of inexpensive old-PC-friendly arcade games. (This example assumes that information about game prices and reviews is also available in the database.) Relational databases thus provide a more modular encoding of the information.

Unfortunately, conventional databases assume information is stored locally, in a consistent format—not externally, in diverse formats, as is the case with information on the Web. Hence they do not solve the problem of organizing information on the Web. To use modular, maintainable representations for information, while still exploiting the power of the Web—its distributed nature, large size, and broad scope—we need practical ways of integrating information from diverse sources.

Why integrating information is hard

Unfortunately, integrating information from multiple sources is very hard. One difficulty is programming a computer to understand the various information sources well enough to answer questions about them. Surprisingly, this is often difficult even when information is presented in sim-

ple, easy-to-parse regular structures such as lists and tables.

As an example, Figure 4 shows a tabular representation of the information in two hypothetical Web sites. Consider the knowledge an integration system would need to answer the following question using these information sources:

> Who publishes "Escape from Dimension Q" and where is their home page?

(In this essay, we assume that questions are given to the information-integration system in a formal language; for readability, however, we'll paraphrase questions in English whenever possible.)

To answer this question, the system must have knowledge of several kinds:

- It must know where to find these tables on the Web, and how they are formatted (*access knowledge*).
- It must know that each tuple $\langle x,y \rangle$ in the table *Website-1* should be interpreted as the statement "the company *y* publishes the game *x*," and that each tuple $\langle t,u \rangle$ in the table *Website-2* should be interpreted as the statement "the home page for the company *t* is found at the URL *u*" (*semantic knowledge*).
- Finally, it must know that the string "Headbone" in *Website-2* refers to the same company as the string "Headbone Interactive" in *Website-1* (*object-identity knowledge*).

Even given all this knowledge, many interesting technical problems remain; however, the technical difficulties involved in using these types of knowledge pale beside the practical difficulties of acquiring the knowledge. Currently, all this knowledge must be manually provided to the integration system and updated whenever the original information sources change. Performing information integration is thus extremely knowledge-intensive and hence expensive in terms of human time.

Of course, many of these problems can be "assumed away:" integrating information sources is not nearly so difficult if they use common object identifiers, adopt a common data format, and use a known ontology. Unfortunately, few existing information sources satisfy these assumptions. The vast majority of existing online sources are designed to communicate only with human readers, not with other programs. We believe that this will continue to hold true, simply because presenting information to a human audience is less demanding for the information provider—information intended for a human audience need not conform to some externally set formal standard; it only has to be comprehensible to a reader.

(a)

GAME TITLE	PUBLISHER
Aladdin Activity Center	Disney Interactive
Arthur's Computer Adventure	Living Books/Broderbund
Escape from Dimension Q	Headbone Interactive
How the Leopard Got His Spots	Microsoft Kids
⋮	⋮

(b)

GAME PUBLISHER	HOME PAGE
Disney	http://www.disneyinteractive.com
Headbone	http://www.headbone.com
Humongous	http://www.humongous.com
Broderbund	http://www.broderbund.com
Microsoft	http://www.microsoft.com
⋮	⋮

Figure 4. Two typical information sources: (a) Web site 1 and (b) Web site 2.

The Whirl approach to information integration

We have written a system for information integration called Whirl. The approach to integration embodied in Whirl is based on two premises:

- *It is unreasonable to assume that all the knowledge needed for information integration will be present*, and in any case impractical to encode this information explicitly. Consequently, inferences made by an integration system are inherently incomplete and uncertain. As in machine learning, speech recognition, and information retrieval, the integration system will have to take some chances and make some mistakes. An integration system thus must have ways of reasoning with uncertain information, and communicating to the user its confidence in an answer.
- *Information integration should exploit the existing human-oriented interface to information sources as much as possible*. It should, whenever possible, understand information using general techniques, analogous to the ones people use, rather than relying on externally provided, problem-specific knowledge. For instance, people have no difficulty recognizing the structures in Table1 as two-column tables; thus a good information-integration system should also be able to recognize such structures. (Although general table-recognition methods exist,[2] to our knowledge, no existing Web-based integration system uses them.) Similarly, most people would judge it likely that the "Headbone" and "Headbone Interactive" denote the same company (or closely related ones), but would consider it unlikely that "Disney Interactive" and "Microsoft" do; an integration system should be able to make a similar judgement, even without knowledge of the domain.

Of the many mechanisms required by such an integration system, we have chosen to concentrate (initially) on general methods for integrating information without object identity knowledge. In most integration tasks, far more object-identity knowledge is needed than any other type of knowledge; while semantic knowledge and access knowledge might be needed for each source, a system potentially needs object-identity knowledge about each pair of constants in the integrated database.

Our approach for dealing with uncertain object identities relies on the observation that information sources tend to use textually similar names for the same real-world objects. This is particularly true when sources are presenting information to people of similar background in similar contexts. To exploit this, the Whirl query language allows users to formulate SQL-like queries about the similarity of names. Consider again the tables of Figure 4. Assuming that the table Website-1 is encoded as a relation with schema `game(name, pubName)`, and Website-2 is encoded as a relation with schema `publisher(name, homepage)`, the question

Table 1. Output of a Whirl query pairing paragraphs of free text and names of computer games. The score is the similarity of the last two columns, normalized to a range of 0–100, and the checkmark indicates if the pairing is correct.

SCORE	DEMO.NAME	GAME.NAME	
80.26	Ubi Software has a demo of Amazing Learning Games with Rayman.	Amazing Learning Games with Rayman	√
78.25	Interplay has a demo of Mario Teaches Typing. (PC)	Mario Teaches Typing	√
75.91	Warner Active has a small interactive demo for Where's Waldo at the Circus and Where's Waldo? Exploring Geography. (Mac and Win)	Where's Waldo? Exploring Geography	√
74.94	MacPlay has demos of Marios Game Gallery and Mario Teaches Typing. (Mac)	Mario Teaches Typing	√
71.56	Interplay has a demo of Mario Teaches Typing. (PC)	Mario Teaches Typing 2	√
68.54	MacPlay has demos of Marios Game Gallery and Mario Teaches Typing. (Mac)	Mario Teaches Typing 2	√
68.45	Psygnosis has an interactive demo for Lemmings Paintball. (Win95)	Lemmings Paintball	√
65.70	ICONOS has a demo of What's The Secret? Volume 1. (Mac and Win)	What's the Secret?	√
64.33	Fox Interactive has a fully working demo version of the Simpsons Cartoon Studio. (Win and Mac)	Simpsons Cartoon Studio	√
62.90	Gryphon Software has demos of Gryphon Bricks, Colorforms Computer Fun Set—Power Rangers and Sailor Moon, and a FREE Gryphon Bricks Screen Saver. (Mac and Win)	Gryphon Bricks	√
60.30	Vividus Software has a free 30 day demo of Web Workshop (Web-authoring package for kids!). (Win 95 and Mac)	Web Workshop	√
59.96	Conexus has two shockwave demos—Bubbleoids (from Super Radio Addition with Mike and Spike) and Hopper (from Phonics Adventure with Sing Along Sam).	Super Radio Addition with Mike & Spike	√

> Who publishes "Escape from Dimension Q" and where is their home page?

might be encoded as the Whirl query

```
SELECT   publisher.name,
         publisher.homepage
FROM     game, publisher
WHERE    (game.pubName ~ game.
         name AND game.name ~
         "Escape from Dimension Q")
```

Here ~ is a similarity operator, and thus the query asks Whirl to find a tuple $\langle u,v \rangle$ from `publisher` such that for some tuple $\langle x,y \rangle$ from `game`, y is textually similar to u, and x is textually similar to the string "Escape from Dimension Q." Such a pair is a plausible answer to the query, although not necessarily a correct one.

This query language is central to our approach, so we will describe it in some detail. The query language has a "soft" semantics; the answer to such a query is not the set of all tuples that satisfy the query, but a list of tuples, each of which is considered a plausible answer to the query and each of which is associated with a numeric score indicating its perceived plausibility.

The universe of possible answers is determined by the `FROM` part of the query; in the example above, possible answers come from the Cartesian product of the `game` and `publisher` relations. Whirl scores answers according to how well they satisfy the conditions in the `WHERE` part of the query: each similarity condition gets a score between zero and one, each Boolean condition receives a score of either zero or one, and Whirl combines these primitive scores as if they were independent probabilities to obtain a score for the entire `WHERE` clause. For the query above, the score of a tuple $\langle u,v,x,y \rangle$ is the product of the similarity of y to u and the similarity of x to "Escape from Dimension Q."

In typical use, Whirl returns only the K highest-scoring answers, where K is a parameter set by the user. From the user's perspective, interacting with the system is thus much like interacting with a search engine: the user requests the first K answers, examines them, and then requests more if necessary.

Semantically, then, the Whirl query language is quite simple—but as in many enterprises, "the devil is in the details." To make this idea work well, we needed to adopt ideas from several research communities. Our system computes the similarity of two names using *cosine distance* in the *vector space model*, a metric widely used in statistical-information retrieval.[1] Roughly speaking, two names are similar according to this metric if they share *terms*, where a term is a word stem, and names are considered more similar if they share more terms, or if the shared terms are rare. As an example, "Disney Interactive" and "Disney" would be more similar than "Disney Interactive" and "Headbone Interactive," because "Interactive" is a more common term than "Disney." These similarity metrics are not well understood formally, but are well supported experimentally.

Whirl also builds on ideas from artificial intelligence. To find the best K answers to a query, we use a variant of A* search,[3,4] coupled with inverted-index techniques developed in the information-retrieval community.[5] In combination, these techniques allow Whirl to find the best K answers to a query fairly quickly, even when the universe of possible answers is extremely large.

What Whirl has accomplished

Using a search-engine-like interface (in which possible answers come in a ranked list) lets us evaluate Whirl in the same way that information retrieval researchers evaluate search engines. In particular, given information about which of Whirl's proposed answers are correct, we can evaluate Whirl using metrics such as recall and precision. We evaluated Whirl on a number of benchmark problems from several different domains, using the measure *of noninterpolated average precision*. (Roughly speaking, this averages the best level of precision obtained at each distinct recall level. The highest possible value for this measure is 100%.) We discovered that the off-the-shelf similarity metric we adopted is surprisingly accurate. On 14 of 18 benchmark problems, average precision is 90% or higher; on seven of the 18 problems, average precision is 99% or higher.

Intriguingly, good performance can often come even when the names from one or both sources are embedded in extraneous text. Table 1 presents the first few answers for the query

```
SELECT      demo.name,game.name
FROM        demo,game
WHERE       demo.name ~ game.name
```

for a problem in which the names in `demo` are embedded in arbitrary paragraph-long passages of free text. As the table's last column shows, most top-ranked pairings are

correct, and the complete ranking of answers Whirl proposed has a respectable average precision of 67%. Whirl's robustness to extraneous noise words means that we can afford to use approximate methods of extracting of data from information sources.

We have also built several nontrivial integrated-information systems using Whirl. The domain of the first is children's computer games. This application integrates information from 16 Web sites. Using the HTML form interface shown in Figure 5, users can construct questions like the following:

> Help me find reviews of games that are in the category "art," are recommended by two or more sites, and are designed for children six years old.

The application knows how to find reviews, demos, and vendors of games, and also understands several properties of games, such as which games are popular and who publishes which games.

Figure 5. The interface to an information-integration system based on Whirl.

We have built a similar system that integrates information about North American birds. Collectively, the integrated databases contain about 100,000 tuples, about 10,000 of which point to external Web pages. Both systems are available on the Web (at *http://whirl.research.att.com/cdroms/* and *http://whirl.research.att.com/birds/*). The response time for complex queries is typically less than 10 seconds. (These time measurements are on a lightly loaded Sun Ultra 2170 with 167-MHz processors and 512 Mbytes of main memory. The current server is not multithreaded, so response times vary greatly with load.)

In building these applications, we deliberately sidestepped many of the problems that have historically been research issues in information integration. Whirl data is not semistructured, [6] but instead is stored in simple relations. The problem of query planning[7] is simplified by collecting all data with spiders offline. We map individual information sources into a global schema using manually constructed views, rather than using more powerful methods.[8,9] Access knowledge is represented in hand-coded extraction programs,[10] rather than learned by example as proposed by Nicholas Kushmeric and others.[11–13] We made these decisions to highlight the advantages of our approach, relative to earlier integration methods: by adopting an uncertain approach to integration, significant applications can be built, even without the aid of other advanced techniques.

The two implemented integration applications illustrate several other important points. First, in the games application, Whirl extracts information about the age range for which games are appropriate from a commercial site; this information can then be used to access a collection of reviews taken from several consumer-oriented sites. The age-range information has, in some sense, been made portable; it has been disassociated from the site that provided it and used for a goal different from its intended purpose (of improving access to a large online catalog). Attaining this sort of modularity and portability of information was one of Whirl's primary goals.

Second, integration need not require complex query interfaces to be useful. As well as providing a query-based interface, the bird application allows data about birds to be browsed, either geographically or by scientific order and family. This browsing interface extends the capability of the original sources, which are seldom organized along both of these dimensions. The bird application also includes a quick-search feature, in which the user types in the name of a bird and gets a list of URLs in response. As an example, in response to a quick search for "great blue heron," Whirl's answer includes a picture indexed only as *ardea herodias*, the scientific name of the great blue heron. This sort of intelligent behavior is enabled by translating the user's quick search into a structured query that exploits an information source giving the scientific nomenclature for birds.

The future of integration

Whirl's current implementation could be extended in many ways. Challenging technical issues include scaling up to larger data sets (the current implementation is memory-based), finding more flexible and more automatic extraction methods, learning to improve scores based on feedback of various kinds, and collecting data efficiently at query time. We will conclude, however, with some more general remarks.

One possible goal for computer science research is the construction of an information system with size and scope comparable to the Web, but with abilities comparable to a knowledge base. In particular, we would like a system that can reason about and understand information that, like information found on the Web, is constructed and maintained in a decentralized fashion. This is a very hard problem and perhaps a very distant goal; however, Whirl represents an important step toward that goal.

Previous systems that access information from multiple sources fall into two main classes. Search engines provide weak and relatively unstructured access to a large number of sites. Previous Web-based information integration systems provide better access to a small number of highly structured sites. Whirl's emphasis on inexact, uncertain integration provides an intermediate step between these two extremes.

The intermediate step of cheap, approximate information integration is a critical one. It is unreasonable to expect an information provider whose primary audience is people to spend much time and energy in making his or her information programmatically available unless there is a clear and immediate benefit. Unfortunately, while integration does provide a benefit, this benefit does not materialize until a number of sources are integrated. In economic terms, the value of having a well-structured, easily-integrated information source is largely external, leading to a classic chicken-and-egg problem. The availability of cheap, approximate integration methods could help to overcome this problem.

Let us close with an analogy. Information integration can be viewed as the problem of getting information sources to talk to each other. Our approach can be viewed as getting information sources to talk to each other in an informal way. We hope that this kind of informal communication will retain much of the utility of formal communication, but be far easier to attain—just as informal essays like this one can, without tedious technical detail, communicate the essence of a new technical result. ■

References

1. G. Salton, ed., *Automatic Text Processing*, Addison Wesley, Reading, Mass., 1989.
2. D. Rus and D. Subramanian, "Customizing Capture and Access," *ACM Trans. Information Sys.*, Vol. 15, No. 1, 1997, pp. 67–101.
3. N. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1987.
4. J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.
5. W.W. Cohen, "Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity," *Proc. ACM Sigmod-98*, ACM Press, New York, 1998, pp. 201-212.
6. D. Suciu, ed., *Proc. Workshop on Management of Semistructured Data*; http://www.research.att.com/~suciu/workshop-papers. html.
7. J.L. Ambite and C.A. Knoblock, "Planning by Rewriting: Efficiently Generating High-quality Plans," *Proc. 14th Nat'l Conf. AI*, AAAI Press, Menlo Park, Calif., 1997, pp. 706–713.
8. A.Y. Levy, A. Rajaraman, and J.J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions," *Proc. 22nd Int'l Conf. Very Large Databases (VLDB-96)*, Morgan Kaufmann, 1996, pp. 251–262.
9. O.M. Duschka and M.R. Genesereth, "Answering Recursive Queries Using Views," *Proc. 16th ACM Sigact-Sigmod-Sigart Symp. Principles of Database Systems (PODS-97)*, ACM Press, 1997, pp. 109–116.
10. W.W. Cohen, "A Web-Based Information System that Reasons with Structured Collections of Text," *Proc. Second Int'l Conf. Autonomous Agents*, ACM Press, New York, 1998, pp. 400–407.
11. N. Kushmerick, D.S. Weld, and R. Doorenbos, "Wrapper Induction for Information Extraction," *Proc. 15th Int'l J. Conf. AI*, AAAI Press, 1997, pp. 729–735.
12. C.A. Knoblock et al., "Modeling Web Sources for Information Integration," *Proc. 15th Nat'l Conf. AI (AAAI-98)*, AAAI Press, 1998, pp. 211–218.
13. C.-N. Hsu, "Initial Results on Wrapping Semistructured Web Pages with Finite-State Transducers and Contextual Rules," *Papers from the 1998 Workshop on AI and Information Integration*, AAAI Press, 1998, pp. 66–73.