Crossing the Structure Chasm

Alon Halevy Oren Etzioni AnHai Doan[*] Zachary Ives[†] Jayant Madhavan
Luke McDowell Igor Tatarinov

University of Washington
{alon, etzioni, anhai, zives, jayant, lucasm, igor}@cs.washington.edu

Abstract

It has frequently been observed that most of the world's data lies *outside* database systems. The reason is that database systems focus on *structured* data, leaving the unstructured realm to others. The world of unstructured data has several very appealing properties, such as ease of authoring, querying and data sharing. In contrast, authoring, querying and sharing structured data require significant effort, albeit with the benefit of rich query languages and exact answers.

We argue that in order to broaden the use of data management tools, we need a concerted effort to cross this *structure chasm*, by importing the attractive properties of the unstructured world into the structured one. As an initial effort in this direction, we introduce the REVERE System, which offers several mechanisms for crossing the structure chasm, and considers as its first application the chasm on the WWW. REVERE includes three innovations: (1) a data creation environment that *entices* people to structure data and enables them to do it rapidly; (2) a data sharing environment, based on a *peer data management system*, in which a web of data is created by establishing local mappings between schemas, and query answering is done over the transitive closure of these mappings; (3) a novel set of tools that are based on computing statistics over corpora of schemata and structured data. In a sense, we are trying to adapt the key techniques of the unstructured world, namely computing statistics over text coropra, into the world of structured data. We sketch how statistics computed over such corpora, which capture common term usage patterns, can be used to create tools for assisting in schema and mapping development. The initial application of REVERE focuses on creating a web of structured data from data that is usually stored in HTML web pages (e.g., personal information, course information, etc.).

1 Introduction and Motivation

Online information comes in two flavors: unstructured corpora of text on one hand, and structured data managed by databases and knowledge bases on the other. These two different kinds of data lead to very different authoring, search and sharing paradigms. In the first, search is based on keywords and answers are *ranked* according to relevance. In the second, search is based on queries in a formal language (e.g., SQL), and all the answers returned for the query are correct according to the underlying semantics of the system. In the U-WORLD of unstructured data, authoring data is straightforward. In contrast, in the S-WORLD of structured data, authoring data is a conceptual effort that requires technical expertise and substantial up-front effort; the author is required to provide a comprehensive structure (i.e., schema) of the domain before entering data.

This paper focuses on the profound difference between the U-WORLD and the S-WORLD: we argue that there is a *structure chasm* between these two worlds. *Crossing this structure chasm* means introducing technology that imports some of the attractive properties of the U-WORLD into the S-WORLD, facilitating the creation of large-scale data sharing systems. Our first goal is to try to place the problem of crossing the chasm prominently in the data management community's agenda. While many research efforts have addressed specific aspects of the problem, we introduce a paradigm that places these efforts in context. Our second goal, which occupies the bulk of the paper, is to provide a set of mechanisms that address different aspects of crossing the chasm. These mechanisms are embodied in the REVERE System, which focuses on the chasm present on the World Wide Web.

1.1 The Structure Chasm

We begin by discussing the key differences between the U-WORLD and S-WORLD in more detail.

1. Authoring: in the U-WORLD, authoring is conceptually straightforward. It is a matter of writing coherent natural language text. In the S-WORLD, authoring is much more complex. One needs to first conceptually organize the data into a schema (or some domain model in a knowledge representation language), before any data can be entered. Designing the schema (even ignoring its physical aspects) is a major undertaking in any real-world domain. Many potential customers of S-WORLD tools are lost immediately: they do not want to or simply cannot create a model of their domain. Those who proceed must invest a significant amount of effort, and they want some return on their investment.

2. Querying: in the U-WORLD, a user need not know much in order to query a collection of data. A set of keywords suffices, and even if those are not the exact words used by the authors, the system will typically still find relevant documents using techniques such as stemming. In the S-WORLD, a user needs to know the precise schema of the data she wishes to query — otherwise, the query will fail. There is no graceful degradation: if a query is not completely appropriate for the schema,

[*] Currently at Department of Computer Science, University of Illinois, Urbana-Champaign.

[†] Present address: Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104.

the user will get no answers. This means that a user needs to understand how *someone else* structured the data, which is often a difficult task.

3. Sensitivity to change: the U-WORLD is relatively insensitive to change. If an author rewords phrases in a document or adds to them, the user does not need to change his queries. In contrast, in the S-WORLD, certain changes to the schema of the data may completely invalidate the queries running against the system. In many cases, this will require significant changes to applications using the database. Again, there is no graceful degradation here: the S-WORLD is brittle in this sense.

4. Ease of sharing data: as a consequence of the difficulties in authoring and querying data, sharing and integrating data is very challenging in the S-WORLD. In the U-WORLD all documents can simply be combined within the same corpus (or indexed by the same search engine), and they can be queried uniformly. In the S-WORLD, because of different domains and tastes in schema design, different data sources are unlikely to use the same schema constructs to represent the same concepts. Thus we must mediate between different schemas (or ontologies): we need to define relationships between data providers and map queries back and forth between them. This is a major effort and typically requires an understanding of both schemas.

5. Accuracy of answers: on the flip side, we can pose much richer queries in the S-WORLD, and the answers returned are exact. The semantics of the underlying system defines a Boolean condition for every candidate answer — it is either true or false. This allows one to automate many tasks and build applications (e.g., managing bank accounts, reserving flights, purchasing books on Amazon, and soon, perhaps, making an appointment with the local dentist via web services). In the U-WORLD, answers are approximations based on expected relevance, and they are ultimately evaluated by a human. Hence, they are useful only in applications where the answers go *directly* to a user who sifts through them "by hand." The approximate nature of the U-WORLD does not mesh well with our expectations from the S-WORLD — we are seldom happy with approximate, incomplete, or incorrect answers. The vast majority of the applications in the S-WORLD simply do not tolerate such answers (and neither do the end users).

Generally, we find that the S-WORLD has more desirable query capabilities, but the U-WORLD enables more rapid and natural content creation, with an added benefit of a quick payoff for the content author. This is why the "average" user produces content in a text file, spreadsheet, or HTML document — but typically avoids relational databases.

The points of this section come as no surprise to anyone who has worked with structured data. The surprising thing is how often these issues are forgotten, especially when people try to design large-scale data sharing systems (e.g., the *Semantic Web* [5]). The problems of creating and sharing structured data are extremely challenging and deeply ingrained in the way people think of structured data. Even sharing structured information within a single, large organization, where presumably everyone is working for the same cause, is known to be a very difficult problem.

1.1.1 Crossing the Chasm

Our goal is to build tools for the S-WORLD that import the attractive properties of the U-WORLD. If we can build tools that make structured content creation, sharing, and maintenance intuitive, fast, and rewarding, then users will be motivated to provide the structural information needed by S-WORLD tools. We do not expect that managing data in the S-WORLD will ever be as easy as in the U-WORLD, but we feel that much of the chasm is an artifact of current data management tools and techniques, rather than the results of inherent differences between the U-WORLD and S-WORLD. We note that crossing the structure chasm does *not* mean merely combining structured and unstructured data in a single system: the two kinds of data already coexist in documents (e.g., [3]), although their coexistence is far from seamless: disparate operations are applied to the different parts.

To illustrate the benefits of crossing the chasm, we now turn to a hypothetical example that runs through the paper.

Example 1.1 Imagine DElearning, an on-line education company, which leverages existing distance learning courses at different universities and weaves them into its own educational programs. A customer of DElearning could take an introductory ancient history course at Berkeley, followed by an intermediate course at Cornell, and culminating in a graduate seminar at Oxford. DElearning pays for the right to send students to different courses, but charges its customers a premium for creating coherent specialized programs that suit their educational needs, schedule constraints, etc. DElearning's strategy for dominating the global distance education market is twofold. First, it plans to rapidly grow its inventory of courses by making it easy for non-technical educators to include their distance learning courses. Second, it seeks to make tailoring a custom educational program easy for potential customers.

Note that neither the U-WORLD nor the S-WORLD offers technology that can meet DElearning's requirements. U-WORLD technology makes joining DElearning easy for educators: they would only need to point DElearning at the URLs for their course web sites. However, searches of HTML pages by potential customers are a tedious way to try and build a custom curriculum. Customers would find that they need to manually check requirements, text books, homework assignments, and schedules, and they would have to do so across HTML pages constructed using different languages, and vocabularies. S-WORLD technology using a global mediated schema would alleviate these problems, but only at the prohibitive up-front cost of authoring a schema that would cover a large number of universities and departments internationally. □

1.2 Overview of REVERE

REVERE (Figure 1) is a highly distributed data management system that addresses different aspects of the structure chasm on the Web. Together, these components can be used to build large-scale data sharing systems. Initially, the goal of REVERE is to build a web of structured data from data that is currently embedded in HTML pages, but which could be used in numerous novel applications if it were available in structured form. REVERE consists of the following components.

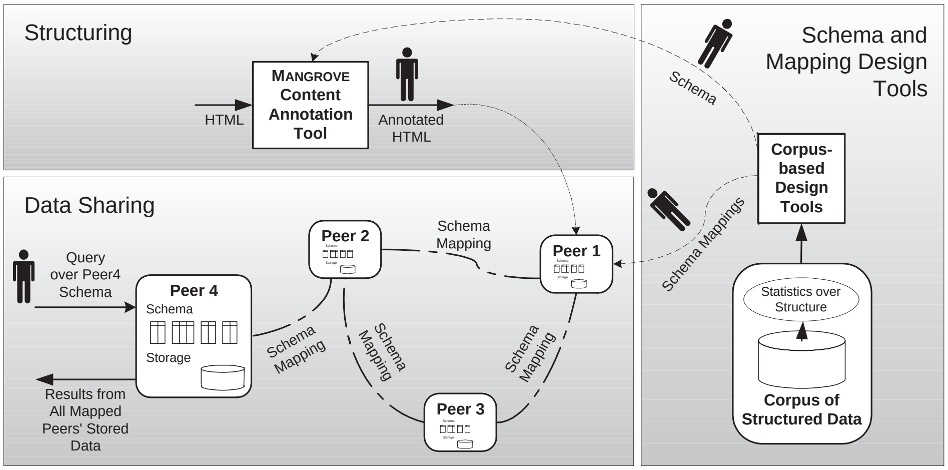
Structuring
Mangrove Content Annotation Tool
HTML
Annotated HTML
Schema
Schema and Mapping Design Tools
Corpus-based Design Tools
Data Sharing
Peer 2
Schema Mapping
Peer 1
Query over Peer4 Schema
Peer 4
Schema
Storage
Schema Mapping
Schema Mapping
Schema Mapping
Schema Mappings
Statistics over Structure
Corpus of Structured Data
Results from All Mapped Peers' Stored Data
Peer 3

Figure 1: The Revere system consists of tools for annotating or structuring existing data; a peer-to-peer data sharing environment in which users can pose queries in any of the peers' schemas, and receive answers from all peers; and tools for defining schemas and mappings, which make use of a corpus of structured data to advise and assist the user. Points where the system interacts with a human are identified with a person icon.

1. The Mangrove data structuring component: The first hurdle in building any large-scale data sharing system is to structure the existing data. The top of Figure 1 illustrates the Mangrove component, which facilitates (and *motivates*) authoring of semantically tagged content *locally*. In the case of Revere, much of the data that we focus on (e.g., contact information, course scheduling, publications, etc.) already resides in HTML pages, and the challenge is to entice users to take the effort to structure the data. The key ideas underlying Mangrove are to try to replicate the principles behind the Web that we believe made HTML authoring explosively popular. Specifically, Mangrove provides (1) a tool for easily annotating unstructured data without having to replicate it, (2) a set of applications that provide instant gratification for authors of structured data and therefore fuel the life cycle of data creation, and (3) deferral of integrity constraints - these are enforced later (to different degrees) by applications using the data. In data-management terms, Mangrove is addressing a scenario where the data comes *before* the schema. In order to entice people to structure their data, we offer a set of lightweight schemas to which they can map their data easily.

2. The Piazza peer-data management system: after data has been locally structured, it needs to be shared with other institutions. The second component of the system (bottom of Figure 1) is a *peer data management system* (PDMS) that enables data to be developed, mapped, and managed in a decentralized and ad hoc fashion. In a PDMS, peers can serve as data providers, logical mediators, or mere query nodes. Semantic mappings between disparate schemas are given *locally* between two (or a small set of) peers. Using these semantic mappings transitively, peers can make use of relevant data anywhere in the system. Consequently, queries in a PDMS can be posed using the local schema of the peer, without having to learn the schema of other peers. We note that PDMSs are a natural step beyond data integration systems, where queries are formulated via a global mediated schema, and all peers must provide mappings from their schemas to this mediated schema. In fact, a PDMS allows for building data-integration and warehousing like applications *locally* where needed.

3. Tools using statistics over structures: the third component of our system (right side of Figure 1) is a tool that facilitates the data authoring and sharing tasks sketched earlier. This tool is based on a corpus of schemas and structured data, and its goal is to extract from this corpus statistical information on *how terms are used in structured data*. In a sense, we are adapting the Information Retrieval paradigm, namely the extraction of statistical information from text corpora, to the S-World. Our hypothesis is that tools built over a corpus of structured data with associated statistics can alleviate some of the key bottlenecks associated with distributed authoring, querying and sharing of structured data. For example, while authoring data, a corpus-tool can be used as an *auto-complete* tool to suggest more complete schemas to a user. When mapping data from different peers in a PDMS, another tool can be used to propose semantic mappings between schemas.

Deploying Revere for DElearning: Revere is ideally suited as a platform for DElearning. It enables a potential customer to inquire about courses, requirements, schedules from any Revere node. Moreover, the query can use the familiar vocabulary of that node and rely on the system to automatically translate the query (and results) appropriately. Revere also makes the addition of a new university or university department into DElearning's network as painless as possible. Since any university already has a course web site, all it must do is use the three components of Revere to join DElearning's "inventory." First, the university's instructors mark up (and periodically update) their course content using Revere mark-up tool. Second, the university's distance learning specialist relies on the Revere corpus to identify peer universities whose schemas are "semantically close" to his own. Finally, the specialist relies on the Revere PDMS to fully specify the exact pairwise mapping between the two universities' schemas. That is all that is required.

The next three sections detail each of the components of Revere.

2 Creation of Structured Data

One of the fundamental barriers to crossing the chasm, which we address with the MANGROVE component of REVERE, is the difficulty in enabling and enticing non-technical content creators to author structured data. As stated earlier, the goal of MANGROVE is to create a semantic web from data that is already in HTML web pages. Dealing this data also raises a challenge of scalability: how do we build a scalable system that provides *efficient* access to a large collection of diverse web pages. The principles underlying MANGROVE are that (1) authors should not be required to duplicate data that already appears in HTML pages, (2) authoring data is local, incremental, and distributed across many people, and (3) authors will be motivated to structure their data *en masse* only if they experience *instant gratification* from services or applications that consume the structured data and immediately produce tangible results.

2.1 Annotating data in MANGROVE

The MANGROVE module enables people to mark-up their data in its current place using a convenient graphical annotation tool. The tool displays a rendered version of the HTML document alongside a tree view of a schema according to which the page is being annotated. Users highlight portions of the HTML document, then annotate by choosing a corresponding tag name from the schema. When the user has finished annotating her HTML document, she uses the tool to *publish* her content, as we described shortly.

The annotations given by the user are embedded in the HTML files but invisible to the browser. This method both ensures backward compatibility with existing web pages and eliminates inconsistency problems arising from having multiple copies of the same data. Our annotation language is syntactic sugar for basic RDF [29]. The reason we had to use a new language is that RDF would require us to replicate all the data in the HTML, rather than supporting in-place annotation.

Schema in MANGROVE: Users of MANGROVE are required to adhere to one of the schemas provided by the MANGROVE administrator at their organization. For instance, a university or department would provide an appropriate domain-specific schema for their users, which may be borrowed or adapted from a schema developed elsewhere. (Section 4 describes tools to assist with this schema creation.) This use of predefined schemas provides simplicity and tractability for the initial development of structured data and associated applications, while still allowing local variations in data expression. Note, however, that MANGROVE users are only required to use a set of standardized tag names (and their allowed nesting structure); as we explain in Section 2.3, they are not required to adhere to integrity constraints (which are often viewed as part of a database schema). Hence, users are free to provide partial, redundant, or conflicting information, which simplifies the process of annotating HTML pages that were originally designed without an agreed-upon schema in mind.

Manual annotation vs. automated wrappers: it is important to note that wrapper technology (e.g., [15, 38]) for automatically extracting structured data from HTML is not adequate for MANGROVE. Wrappers rely on the existence of many web pages with very similar structure, whereas in our case, we have many pages with very differing structures. Similarly, information extraction techniques (e.g., [44]) are inadequate here as they are based on domain-specific heuristics and are often unreliable.

2.2 Instant Gratification Applications

In the U-WORLD, rendering a new HTML page on the browser enables the user to immediately see the fruits of her labor. Adding links enables her to immediately participate in a web of data. MANGROVE tries to replicate these conditions for editing S-WORLD data. *Instant gratification* is provided by building a set of applications over MANGROVE that immediately show the user the value of structuring her data. For example, an online department schedule is created based on the annotations department members add to course home pages, talk calendars, readings group pages, etc. Other applications that we are constructing include a departmental paper database, a “Who’s Who,” and an annotation-enabled search engine. Thus, when a user explicitly *publishes* new data using REVERE’s graphical tool, the applications are immediately updated, and the user can see her changes take immediate effect. Moreover, this tangible result encourages a feedback cycle where users expand and tweak their documents to get the desired data result, just as users modify traditional HTML documents to achieve the desired visual effects. This feedback cycle would be crippled if changes relied upon periodic web crawls before they took effect.

Storing annotations: creating the instant gratification applications raises another scale-up challenge. A system that would access the HTML content at *query time* would be impractical. The reason is that the number of web pages is huge, and there is no a priori way to prune access to irrelevant documents at query time. This is in contrast to data integration systems [20, 46, 23, 2, 31, 26, 16] and their subsequent commercial incarnations that provide *HTML gateways*, where certain data was left in HTML form, and accessed through a wrapper at query time. Such systems assumed that the contents of the web pages were described to the system, and hence most of them could be pruned at query optimization time.

In MANGROVE, the annotations on web pages are stored in a repository for querying and access by applications. For ease of implementation, we currently store the data in a relational database using a simple graph representation. We use the Jena [33] RDF-based querying system in order to pose RDF-style queries. The database is typically updated the moment a user *publishes* new or revised content.

2.3 Treatment of Integrity Constraints

The third way in which MANGROVE mimics the authoring conditions on the WWW is that it frees authors from considering integrity constraints. When a person edits her home page to add her phone number, she does not need to check whether her phone number appears elsewhere on the web, and whether the number is consistent. In MANGROVE any author is free to publish any data, regardless of what else is published. Hence, the database created from the web pages may have dirty data: it may be inconsistent; certain attributes may have multiple values, where there should be only one;

there may even be wrong data that was put on some web page maliciously.

The burden of *cleaning* up the data is passed to the application using the data, based on the observation that anyway different applications will have varying requirements for data integrity. In some applications, clean data may not be as important, possibly because users can tell easily whether the answers they are receiving are correct or not (possibly by following an additional hyperlink). For other applications, it may be important that data be very consistent (e.g., that you show only a single phone number), and there may be some obvious heuristics on how to resolve conflicts. For example, if the application is creating a phone directory of the department's faculty, then the application can be instructed to extract a phone number from the faculty's web space, rather than anywhere on the web. The source URL of the data is stored in the database and can serve as an important resource for cleaning up the data. In a sense, using the URL parallels the operation of the web today, where users examine web content and/or its apparent source to determine the usefulness of the content. Finally, in addition to dealing with inconsistent data as necessary, one can also build special applications whose goal is to proactively find inconsistencies in the database and notify the relevant authors. We note that deferring constraint checking to the application is a significant departure from today's practice in data management systems. However, it is necessary if we are going to allow large-scale distributed authoring.

Example 2.1 Returning to our example, we note that data about courses is typically stored in two places. The first is a university database that contains the name of the instructor, hour and location of the course and a course description. The second, and more comprehensive source of data on a course, is its web page. There one finds the instructor's office hours, TA information, textbook, assignments, slides, and all other information that is specific to the particular course offering. The first kind of data is already in structured form; MANGROVE assists with the second collection, which tends to be much less structured. Assume that Tsinghua University desires to make it easier for students and staff to find relevant course information. Tsinghua offers a very large number of courses, and each course already has an existing web page that provides descriptive and schedule information. While it might be possible for the administrator to annotate the web pages of each course, the large number of courses presents a heavy initial and maintenance burden. Instead, the instructors annotate their own course web pages, aided by the MANGROVE annotation tool. Instructors are motivated to annotate (along with perhaps some encouragement from the university!) by seeing their course added to the MANGROVE-enabled applications. Because the annotated information is part of the standard course page, the information will stay current, and it is simple to maintain the proper annotations when the course web page changes. □

While this example focuses upon the annotation of web pages that are constructed by hand, annotations could also be easily added to pages that are generated from a database. Furthermore, MANGROVE also enables some web pages that are currently compiled by hand, such as department-wide course summaries, to be dynamically generated in the spirit of systems like Strudel [17].

2.4 MANGROVE and the Chasm

The MANGROVE module takes several steps towards crossing the structure chasm. First, it enables users to author structured data in a familiar environment while leaving the data where it already is. It provides a convenient tool for annotating pre-existing data. The instant gratification applications entice users to incrementally structure their data — even small amounts of annotation can produce tangible results. As annotations become common, more sophisticated applications will arise to exploit them, which will in turn promote the creation of more structured data.

3 Decentralized Data Sharing

In the previous section, we described how REVERE facilitates authoring of local structured content. This section describes how REVERE supports *large scale data sharing* of structured content across multiple, disparate communities.

Most of the challenges in data sharing arise from one central problem: different sources of data will naturally use different schemas to structure their information. This can arise simply because one content developer has a different way of conceptualizing a domain from others, or it can also arise because two different data sources have somewhat different domains or requirements. In either case, combining data from multiple schemas lies at the heart of sharing structured data.

A commonly proposed approach is the one used by data warehousing [9] and data integration [20, 23, 2, 31, 26]: create a common, *mediated* schema that encompasses the concepts to be shared, and define mappings between each source's schema and the mediated schema. Users can query over the individual data sources' schemas, only getting answers from the local data, or over the mediated schema, getting answers from all sources with mappings. This approach works well enough to be practical for many problems, but it scales poorly, for two reasons. First, the mediated schema can be extremely difficult and time-consuming to create, and also slow to evolve, as all users of the system must agree how the data can be represented and consent to any future changes. Schema creation at the global level is simply too heavyweight for quick data sharing tasks. Second, data providers must learn a new (and often entirely different) schema if they are to actually benefit from the data sharing arrangement. They may not consider the rewards to be worth the effort.

In REVERE, our goal is to provide mediation between schemas in a decentralized, incremental, bottom-up fashion that does not require global standardization, and which does not require users to learn a new schema. The goal of our *peer data management system* (PDMS) component is to create an ad hoc environment in which every participant can add new structured data, new schemas, and new mappings between schemas. Every user of the system can pose a query over her preferred schema. The PDMS will find all data sources related through this schema via the transitive closure of mappings, and it will use these sources to answer the query in the user's schema. Our approach addresses the problems cited above, and it brings many U-WORLD-like capabilities to schema

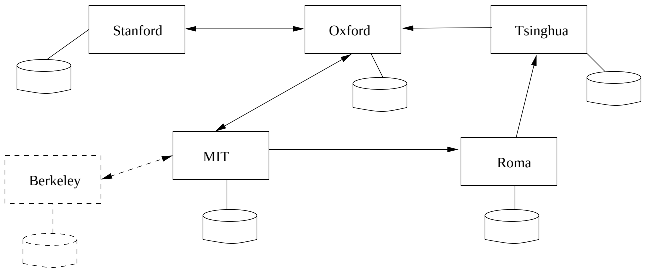

Figure 2: PDMS for our university example. The arrows correspond to schema mappings between peers. No central mediated schema is necessary. As long as the mapping graph is connected, any peer can access data at any other peer by following schema mapping "links".

creation and mediation. We support incremental creation of new schema concepts and new mappings, meaning that each user can easily extend the system, without needing global agreement. We allow users to continue to query using their existing schemas, rather than forcing them to learn new ones, meaning that data sharing becomes nearly automatic and transparent once the appropriate mappings are established.

The natural extensibility of a PDMS can best be illustrated by continuing with our running example. In Section 2, we saw how individual university web sites could be annotated with semantic information.

Example 3.1 Suppose that universities throughout the world have adopted REVERE's content authoring tools and annotated their web pages. These universities also made use of the REVERE query tools to support ad hoc queries from users, and they developed dynamic web pages (e.g., university-wide seminar calendar) from views defined over the structured data.

Now these universities want to join the DElearning network of distance-education courses. Naturally, each university used a different, independently evolved schema to mark up its web pages. For the reasons cited above, creating a single mediated schema for all universities is infeasible. Furthermore, with a mediated schema it is hard to leverage the work done by others — if the University of Rome, that has a schema using terms in Italian, maps its schema to a mediated schema that uses terms in English, this does not help the University of Trento. It would be much easier for Trento to provide a mapping to the Rome schema and leverage their previous mapping efforts.

Peer data management techniques are much more appropriate for this task, as shown in Figure 2. Initially, a few universities define mappings among their schemas, such that they are transitively connected. Now, as other universities agree to join the coalition, they form mappings to the schema *most similar to theirs* (e.g., Trento maps to Rome) — they will be transitively connected to the others. The moment a peer establishes mappings to other sources, it can pose queries using its native schema, which will return answers from all mapped peers. As a result, every participating university will feature the full set of distance-education courses, without having to make any significant modifications to its infrastructure (beyond possibly extending its schema to include a few new concepts such as the language in which each course is taught). A student now can choose courses from all over the world, but all interactions will be done directly through the local university, in as transparent a fashion as possible. □

The example illustrates an important characteristic about mappings in a PDMS. One of the advantages of data integration systems is that the number of semantic mappings we need to specify is only linear in the number of data sources. We emphasize that in a PDMS, we do not need to specify mappings between *every pair* of peers. Instead, every peer specifies a mapping to the most convenient other peer(s). Hence, the number of mappings may still be linear, but peers are not forced to map to a single mediated schema.

The reason we refer to our system as a *peer* data management system is that it not only focuses on ad hoc, decentralized logical extensibility (in which every participant can define its own schema and provide its own data), but we couple that with a flexible, decentralized, peer-to-peer system architecture. Peer-to-peer systems (popularized by file-sharing systems such as Napster and Gnutella, but also the topic of a significant body of research in distributed systems (e.g., [11, 45, 41]) seek to provide a fully decentralized infrastructure in which every participant or peer provides resources to the overall system, resulting in a system that scales well as members are added; and every member can join or leave at will.

Our initial PDMS implementation is a system called Piazza [21, 25], and we now highlight some aspects of the Piazza system architecture.

3.1 System Architecture

Piazza consists of an overlay network of peers connected via the Internet. Each peer may provide new content and services to the overall system, plus it may make use of the system by posing queries. Piazza assumes an XML data model, since this is general enough to encompass relational, hierarchical, or semi-structured data, including marked up HTML pages.

A peer can provide any or all of three different types of content: (1) new XML data (which we refer to as *stored relations*[1] to emphasize the fact that they are materialized source data), (2) a new logical schema that others can query or map to (we refer to this as a *peer schema* or a set of *peer relations*), and (3) new mappings between peer relations and source relations or other peer schemas. A peer's services may include query answering (with respect to its peer schema, or even the schema of another peer), materialization of views (to replicate data for performance or reliability), and potentially storage and processing of meta-information for coordinating the overall PDMS.

3.1.1 Query Answering in a PDMS

The problem at the heart of REVERE's PDMS is that of query answering: every user query is posed over a logical peer schema and must be rewritten so it ultimately refers only to stored relations on the various peers.

In data integration, we find a two-tiered architecture, with a mediated schema defined over a set of data sources. Two classes of formalisms were developed to express relationships between sources and the mediated schema (see [24]): *global-as-view*, in which the mediated schema is defined as a set of

[1] Note that we use the term "relation" in a very loose sense, referring to any flat or hierarchical structure, including XML.

Berkeley peer schema (XML DTD):
Element schedule(college*)
Element college(name, dept*)
Element dept(name, course*)
Element course(title, size)

MIT peer schema:
Element catalog(course*)
Element course(name, subject*)
Element subject(title, enrollment)

Figure 3: Example peer schemas (in XML DTD form) for Berkeley and MIT.

queries over the data sources; and *local-as-view*, in which the data sources are defined as views over the mediated schema.

In Piazza, we find two significant issues that need to be addressed. The first is that our mappings are specified between small subsets of peers, and query answering must be done using the transitive closure of the mappings. The second is that our mapping formalism needs to support querying over XML, rather than conjunctive queries over relations.

Our initial work on query answering in a PDMS [25] addresses the first issue. We examined how the techniques used for conjunctive queries in data integration can be combined and extended to deal with the more general PDMS architecture. The key challenge in query answering is how to make use of the mappings to answer a query. We must extend from the two-tier architecture of data integration to a graph structure of interrelated mappings: a query should be rewritten using sources reachable through the transitive closure of all mappings. However, mappings are defined "directionally" with query expressions (using the GLAV formalism [19]), and a given user query may have to be evaluated against the mapping in either the "forward" or "backward" direction. This means that our query answering algorithm has aspects of both global-as-view and local-as-view: it performs query unfolding and query reformulation using views. In addition, our query answering algorithm is aided by heuristics that prune redundant and irrelevant paths through the space of mappings.

We are now developing a mapping language for relating XML data, and a set of reformulation algorithms to operate over them. (See Figure 3 for an example of peer schemas for the DElearning example). Our mapping language begins with a "template" defined from a peer's schema; the peer's database administrator will annotate portions of this template with query information defining how to extract the required data from source relations or other peer schemas. This approach bears similarities to the XDR mapping representation of Microsoft SQL Server [42] and the annotations used by IBM's XML Extender [49], but we actually use a subset of XQuery to define the mappings from XML data to an XML schema, rather than from relational data to an XML schema. In Figure 4, we see an example of a mapping from Berkeley's peer schema to MIT's schema. We have a preliminary version (and implementation) of the mapping language, which supports hierarchical XML construction and limited path expressions, but avoids most of the complex (and hard-to-reason-about) features of XQuery; our goal is to keep query translation tractable but to support the most common language constructs.

```
<catalog>
  <course> {$c = document("Berkeley.xml")/schedule
                                   /college/dept}
    <name> $c/name/text() </name>
    <subject> { $s = $c/course }
      <title> $s/title/text() </title>
      <enrollment> $s/size/text() </enrollment>
    </subject>
  </course>
</catalog>
```

Figure 4: Berkeley-to-MIT schema mapping. The template matches MIT's schema. The brace-delimited annotations describe, in query form, how variables (prefixed with dollar-signs) are bound to values in the source document; each binding results in an instantiation of the portion of the template with the annotation.

3.1.2 Peer-based Query Processing

The logical schema mapping and query translation facilities discussed above would be sufficient to provide the decentralized data sharing system we desire. One could imagine building a central server that receives a query request made over a particular schema, translates the query to reference only source data, fetches the data, and processes the data according to the query plan. However, this approach does not make good use of the compute and storage resources available across the peers within the PDMS, and it ultimately would become a bottleneck. We would vastly prefer a more Web-like environment in REVERE, in which each peer can receive and process requests — and in which peers can also perform the duties of cooperative web caches [51] and content distribution networks like Akamai.

Thus, a major focus of research in the Piazza system is on distributed query processing and *data placement*. Our ultimate goal is to materialize the best views at each peer to allow answering queries most efficiently, given network constraints; and to distribute each query in the PDMS to the peer that will provide the best performance. However, we must also do this in an environment where the data sources are subject to update at any point, and hence view updates can become expensive.

Propagation of updates is also a major challenge in a PDMS: we would prefer to make incremental updates versus simply invalidating views and re-reading data. Piazza treats updates as first-class citizens, as any other data source, in the form of "updategrams" [36]. Updategrams on base data can be combined to create updategrams for views. When a view is recomputed on a Piazza node, the query optimizer decides which updategrams to use in a cost-based fashion. Ultimately, we want to support updating of data through views, extending the techniques covered in [22].

3.2 Piazza and the Chasm

Piazza contributes to crossing the structure chasm by combining the ad hoc extensibility of the Web with the rich semantics of database systems. The schema is not in one place any longer — it is distributed across many peers and managed by local relationships. In fact, there may not even be a global consistent schema for the entire system.

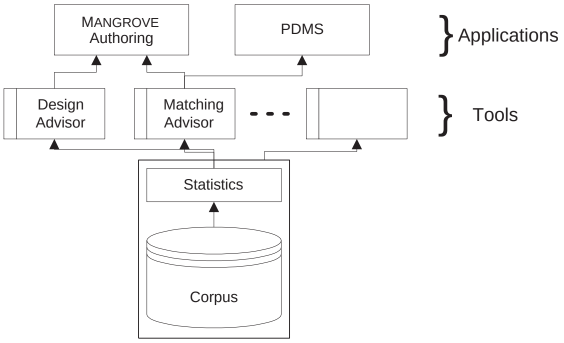

Figure 5: We propose the use of statistical information about structures to alleviate some of the key difficulties of the S-WORLD. The technique is based on collecting corpora (perhaps domain-specific) of structures, and computing a set of statistics on how terms are used in structures. The statistics will be used in a set of general purpose tools that are embedded in various applications.

4 Statistics over Structures

In the previous sections we presented the architecture of two components of the REVERE system that ease the process of authoring and sharing structured data. However, even with these tools, significant design effort is required, e.g., in creating schemas appropriate for mark-up of data, and in creating the mappings that relate different peers' schemas. In this section, we describe a third component that will provide intelligent support to the previously mentioned design tools, thereby significantly reducing the tedium in authoring, querying, and sharing data.

We propose to build for the S-WORLD the analog of one of the most powerful techniques of the U-WORLD, namely the statistical analysis of corpora. A number of techniques in the U-WORLD are based on statistical information on word usage and occurrence patterns in large corpora of text documents. For example, consider the popular TF/IDF [43] (Term Frequency/Inverse Document Frequency) measure. This measure is commonly used to decide the relevance of a document to a keyword query: a document is considered relevant if the number of occurrences of the keyword in the document is statistically significant w.r.t. the number of appearances in an average document. Furthermore, co-occurrences of words in multiple documents can be used to infer the relevance of one word to another. Such document corpora are compiled for specific domains, thereby exploiting the special domain characteristics of word usage.

Our goal is to build corpora of structured data (see Figure 5) from which we will be able to extract extensive statistics about *how data is structured*. Based on these statistics, we will build a set of general purpose tools to assist structuring and mapping applications.

4.1 Corpus of Structures

Each corpus will include:

- forms of schema information: relational, OO and XML schemas (possibly including their associated integrity constraints), DTDs, knowledge-base terminologies (ontologies),
- queries over these schemas and ontologies,
- known mappings between schemas in the corpus,
- actual data: example tables, XML documents, ground facts of a knowledge base,
- relevant metadata that is associated with structured data (e.g., database statistics).

It is important to emphasize that a corpus is *not* expected to be a coherent universal database in the spirit of the Cyc knowledge base [30], which attempts to model all commonsense knowledge. It is just a collection of disparate structures. We expect that the schema information of the corpus will be stored and accessed using tools for model management [7], which provides a basic set of operations for manipulating models of data (e.g., schemas, XML DTDs). However, our goals with the corpus go beyond those of model management – we emphasize the collection of statistical information, and the tools that can be built with these statistics.

4.2 The Statistics

Given the contents of the corpus, there is a plethora of statistics that can be maintained over it. Finding the most effective types of statistics to compute for the corpus is a long term research challenge. In what follows we describe the types of statistics we initially plan to compute and maintain in the corpus. We consider two kinds of statistics: *basic* and *composite*.

4.2.1 Basic Statistics

Basic statistics are associated with words in the English (or for that matter, any) language. Informally, these statistics indicate how a word is used in different roles in structured data. For each of these statistics, we maintain different versions, depending on whether we take into consideration word stemming, synonym tables, inter-language dictionaries, or any combination of these three; the basic statistics include:

Term usage: How frequently the term is used as a relation name, attribute name, or in data (both as a percent of all of its uses and as a percent of structures in the corpus).

Co-occurring schema elements: For each of the different uses of a term, which relation names and attributes tend to appear with it? What tend to be the names of related tables and their attribute names? What tend to be the join predicates on pairs of tables? Are there clusters of attribute names that appear in conjunction? Are there mutually exclusive uses of attribute names?

Similar names: For each of the uses of a term, which other words tend to be used with similar statistical characteristics?

4.2.2 Composite Statistics

Composite statistics are similar to the ones above, but maintained about *partial structures*. Examples of partial structures are sets of data instances, relations with associated attribute names, a relation with some data (possibly with missing values).

Clearly, a key challenge we face is that the number of partial structures is virtually infinite, and we will not be able to maintain all possible statistics. Hence, we will maintain only statistics on partial structures that appear frequently (discovered using techniques such as [50, 18, 39]), and estimate the statistics for other partial structures.

4.3 The Tools

We now describe two interactive tools that will be built using a corpus and associated statistics and used to support components of REVERE.

4.3.1 Authoring Structured Data

The first tool, DESIGNADVISOR, assists with the authoring of structured data, and will be used in MANGROVE. By authoring, we mean any kind of user activity whose end result is a set of structured data.

The idea of DESIGNADVISOR is illustrated in our distance learning example. Suppose the University of Washington is planning to join the DELearning consortium. A coordinator is assigned the task of creating the schema that will be used to publish course offerings. While this would have been a daunting task, the coordinator now proceeds as follows. First, she creates a schema fragment and some data about course names, contents, and instructors. Then, she uses the DESIGNADVISOR to propose extensions to the schema using the corpus. Note that in this case, the set of schemas already in REVERE is an excellent starting point for a useful corpus.

DESIGNADVISOR uses the corpus and its statistics and returns a ranked list of similar schemas. The coordinator chooses a schema from the list and modifies it further to fit the local context. The chosen schema may not completely model what the coordinator requires. For example, it may not model information about teaching assistants (TAs). So the coordinator proceeds to add several attributes such as name and contact-phone to the course table. At this point, DESIGNADVISOR, which has been monitoring the coordinator's actions, steps in and tells the coordinator that in similar schemas at most other universities, TA information has been modeled in a table separate from the course table. The coordinator takes this input and modifies the schema design accordingly.

More concretely, DESIGNADVISOR performs the following function. It is given a *fragment* of a database, i.e., a pair (S, D), where S is a partial schema and D is a (possibly empty) set of data that conforms to S. The tool returns a ranked set of schemas S', where for each S' there is a mapping of S *into* S'. That is, S' models a superset of the attributes and relations in S, but may also modify the way S models the domain. The tool may create the mappings by employing the SCHEMAMATCHER tool which we describe shortly.

DESIGNADVISOR ranks the schemas in the proposed set in decreasing order of their similarity. A general template for a similarity function can be defined as follows:

$$sim(S', (S, D)) = \alpha \cdot fit(S', S, D) + \beta \cdot preference(S'),$$

where α and β are weights on the following terms:

- $fit(S', S, D)$ measures the basic fit for S and S' (i.e., do S and S' model the same domains), and is currently defined to be the ratio between the total number of mappings between S' and S and the total number of elements of S' and S.
- $preference(S')$ incorporates user preference criteria, such as whether S' is commonly used, conforms to a particular set of schemas, or is relatively concise and minimal.

The benefits of a tool such as DESIGNADVISOR are:

1. time savings: similar to other *auto-compete* features, the author can begin to design the schema and immediately be proposed a complete (or near complete) one,
2. better design: instead of the user having to design (and redesign) the schema, the proposed schema may already be one that is known to be well designed, and
3. conformance to standards: the system may be able to guide the user into schemas that conform to standards or otherwise commonly used schemas.

Note that the last case would have to be reflected in the ranking criteria that DESIGNADVISOR uses.

4.3.2 Assisting Schema Matching

The second tool, MATCHINGADVISOR, assists local coordinators in mapping their schemas to others, and hence facilitates creation of the semantic mappings that underlie a PDMS. The general problem of schema matching has recently attracted significant interest in both the DB and AI communities (see [40] for a recent survey). Schema matching is also one of the proposed operations in model management, and hence our MATCHINGADVISOR tool can be viewed as another (yet very different) type of semi-automatic tool for schema matching.

The goal of schema matching is the following. Given two schemas, S_1 and S_2, we want a mapping M that relates the concepts of S_1 and S_2. There are several variants to mappings. In the simple case, a mapping only provides correspondences between terms in S_1 and terms in S_2, whereas in more complex cases, the mapping will include query expressions that enable mapping the data underlying S_1 to S_2 or vice versa. Note that there may be only a partial match between S_1 and S_2, and hence some terms in one may not have corresponding terms in the other. In addition, inputs to the schema mapping problem may also include data instances of one or both schemas.

We illustrate one way of building MATCHINGADVISOR, which extends our previous work on schema matching in the LSD [13] and GLUE [14] Systems. We first briefly recall the main ideas in LSD.

LSD considered the schema matching problem in the context of data integration, where the system exposes a single mediated schema, and we need to provide mappings from the mediated schema to each of the data sources. The idea in LSD was that the first few data sources be manually mapped to the mediated schema. Based on this *training*, the system should be able to predict mappings for subsequent data sources. To do this, LSD uses the information in the manual mappings to learn a classifier for every element in the mediated schema (in our case, a classifier for every tag in the mediated XML DTD). The system uses a multi-strategy learning method that can employ multiple learners, thereby having the ability to learn from different kinds of information in the input (e.g., values of the data instances, names of attributes, proximity of attributes, structure of the schema, etc). The results of applying LSD on some real-world domain show matching accuracies in the 70%-90% range.

The classifiers computed by LSD actually encode a statistic for a composite structure that includes the set of values in a column and the column name. Given such a structure for a new column in a data source, the classifiers return the likelihood that the structure corresponds to a mediated schema element.

We can use these classifiers to build MATCHINGADVISOR, which finds a mapping between two *previously unseen* schemas. Given two schemas, S_1 and S_2, we apply the classifiers in the corpus to their elements respectively, and find correlations in the predictions for elements of S_1 and S_2. For example, if we find that all (or most) of the classifiers had the same prediction on element $s_1 \in S_1$ and $s_2 \in S_2$, then we may hypothesize that s_1 matches s_2.

An alternative way to use the corpus for schema matching is via the DESIGNADVISOR tool. The idea here would be to find two example schemas in the corpus that are deemed by DESIGNADVISOR to be similar to S_1 and S_2, respectively, and then use mappings between those schemas *within* the corpus to map between S_1 and S_2. In general, the corpus and its associated statistics act as a domain expert because numerous existing schemas and schema fragments might be similar to the schemas being matched. This domain expert can be used in a variety of ways to facilitate schema mappings.

4.4 Corpus and the Chasm

Exploiting statistics over structures holds great potential in simplifying many of the hardest activities involved in managing structured data. As discussed, the corpus and its statistics can be used to facilitate authoring structured data (and hence useful in MANGROVE) and discovering semantic mappings between different structures (and therefore useful in creating mappings for Piazza).

Another area where the corpus is relevant to the chasm is in facilitating the querying of unfamiliar data. Specifically, a user should be able to access a database (or set of databases), the schema of which she does not know, and pose a query using *her own* terminology (or possibly using natural language). One can imagine a tool that uses the corpus to propose reformulations of the user's query that are well formed w.r.t. the schema at hand. The tool may propose a few such queries (possibly with example answers), and let the user choose among them or refine them.

5 Related Work

While we believe that the problem of crossing the structure chasm has not previously been addressed in a comprehensive fashion, our line of research clearly builds upon a great deal of work from the database, AI, and information retrieval communities. Because of space limitations, we can only cover these works very briefly.

Clearly, one approach to crossing the chasm is to leave the data unstructured and try to answer S-WORLD queries over it. Answering queries directly over unstructured data is typically very difficult, as it requires natural-language understanding techniques and unambiguous text, but it may work for certain cases, e.g., as in the natural language query answering system of [28].

The current approach to building the Semantic Web [5] is based on annotating the data with ontologies. Ontologies, written in an expressive knowledge representation language (e.g., OWL [12]), enhance the understanding of the data, and therefore facilitate sharing and integration. In contrast, mediation has received very little attention in the Semantic Web community.

The problem of querying structured data, but allowing for approximations or ranked results in the query, has been quite well-studied in the database community (see [8] for a survey of a recent workshop on flexible query answering). Examples include query relaxation [10], cooperative query answering [35], returning approximate and imprecise answers [1, 37], and extensions of SQL or XQuery to support ranked queries over unstructured text fields or elements (e.g., IBM's DB2 Text Extender and [48]). The semantics of these query languages still tend to be biased towards querying structure, but the answers are no longer guaranteed to be correct.

Originally, semi-structured data was touted as the solution to flexible construction of structured data. XML, the most prevalent form of semi-structured data, was intended to ease authoring of data: one did not need to design the schema before entering the data. However, both the uses of XML and the tools and techniques developed for managing XML in recent years have focused exclusively on S-WORLD issues. For instance, XML Schema has been developed to provide richer semantics to XML documents; XQuery and other languages for querying semi-structured data or Web hyperlink structure [34] provide some more flexibility in querying: they support irregularities in the structure, but are still essentially S-WORLD languages. XML is primarily used to exchange data that is encodable in an S-WORLD formalism. The U-WORLD uses of XML mostly relate to using it as a format for exchanging marked-up text documents (e.g., reports or news articles).

The idea of mediating between different databases using local semantic relationships is not new. Federated databases and cooperative databases have used the notion of inter-schema dependencies to define semantic relationships between databases in a federation (e.g., [32]). However, they assumed that each database in the federation stored data, and hence the focus was on mapping between the stored relations in the federation. Our emphasis is on supporting schema mediation among large numbers of peers, so we need to be able to map both relationships among stored relations and among conceptual relations (i.e., extensional vs. intentional relations), and we must be able to quickly chain through multiple peer descriptions in order to locate data relevant to a query.

In [21] we first described some of the challenges involved in building a PDMS. The focus there was on *intelligent data placement*, a technique for materializing views at different points in the network in order to improve performance and availability. In [27] the authors study a variant of the data placement problem, focusing on intelligently caching and reusing queries in an OLAP environment. An alternative language for mediating between peers appears in [6].

In building the corpus-based tools, we expect that a variety of techniques that have been developed for summarizing, mining, and clustering XML [50, 39, 47] will be useful for computing the statistics associated with the corpus. In addition, the idea of using prior mappings to aid in building new ones has been used in [40, 4, 13].

6 Conclusion

At first glance, the structure chasm between the U-WORLD and the S-WORLD seems all-but-bridgeable as a result of the inherent differences between the two worlds — after all, in the U-WORLD, we have insufficient semantic information to provide precise and complete answers, enforce integrity constraints, or combine information in meaningful ways. While this is indeed true, the chasm has actually been widened by tools on the structured data management side, which have made content creation difficult.

Our focus in crossing the structure chasm has been on rethinking the design of S-WORLD tools for content creation and data sharing. We have presented a detailed path for crossing the chasm by developing the REVERE data system, which facilitates the authoring, querying and sharing of data in the S-WORLD. In REVERE, we have begun building a content annotation tool that makes marking up text easy and rewarding, we have developed a peer data management system that mediates between different schemas while providing the user with a familiar schema in which they can pose queries, and we have proposed the use of tools that exploit statistics over corpora of structures to advise and assist schema and mapping designers. Together, these three components of REVERE make it much easier to convert the vast wealth of unstructured content into structured form. Moreover, REVERE enables incremental, bottom-up structuring of data rather than requiring the massive, up-front effort of creating a single, universal schema as a prelude to any data sharing.

While we believe that REVERE is an important step in crossing the chasm, it is also clear to us that the bigger problem — building data management tools that effectively handle the vast body of real-world data, which lies outside the database — is an immense one that requires significant contributions by our entire community (as well as related communities). We would like to conclude by urging others in the database community to take a fresh look at the problems of the chasm, and to see where techniques from the structured world can be extended to be more broadly applicable.

Acknowledgments

We thank Pedro Domingos, Steve Gribble, Hank Levy, Peter Mork, Maya Rodrig, Dan Suciu, Deepak Verma and Stanislava Vlasseva for the contributions to the design and implementation of several components of the REVERE system. Special thanks to Phil Bernstein for many thoughtful discussions. Phil Bernstein, Steve Gribble, Hank Levy, Natasha Noy and Dan Suciu provided excellent comments on earlier versions of the paper. Finally, special thanks to Mike Stonebraker for a careful review of our paper. Some of this work is funded by NSF ITR Grant IIS-0205635 and Grant IIS-9985114.

References

[1] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The Aqua approximate query answering system. In *SIGMOD '99*, pages 574–576, 1999.

[2] S. Adali, K. Candan, Y. Papakonstantinou, and V. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of SIGMOD*, pages 137–148, Montreal, Canada, 1996.

[3] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL. *Nucleic Acids Research*, 28:45–48, 2000.

[4] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, 2002.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[6] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing : A vision. In *ACM SIGMOD WebDB Workshop 2002*, 2002.

[7] P. A. Bernstein, A. Y. Halevy, and R. Pottinger. A vision of management of complex models. *SIGMOD Record*, 29(4):55–63, December 2000.

[8] P. Bosc, A. Motro, and G. Pasi. Report on the fourth international conference on flexible query answerng systems. *SIGMOD Record*, 30(1), 2001.

[9] S. Chaudhuri and U. Dayal. An overview of data warehouse and OLAP technology. *SIGMOD Record*, 26(1), March 1997.

[10] W. W. Chu, M. A. Merzbacher, and L. Berkovich. The design and implementation of CoBase. In *SIGMOD '93*, pages 517–522, 1993.

[11] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *ICSI Workshop on Design Issues in Anonymity, Berkeley, CA*, July 2000.

[12] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL web ontology language 1.0 reference, 2002. Manuscript available from `http://www.w3.org/2001/sw/WebOnt/`.

[13] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.

[14] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Int. WWW Conf.*, 2002.

[15] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 39–48, Marina del Rey, CA, USA, 1997. ACM Press.

[16] D. Draper, A. Y. Halevy, and D. S. Weld. The nimble integration system. In *Proc. of SIGMOD*, 2001.

[17] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Declarative specification of web sites with strudel. *VLDB Journal*, 9(1):38–55, 2000.

[18] J. Freire, J. R. Haritsa, M. Ramanath, P. Roy, and J. Simeon. StatiX: Making XML count. In *SIGMOD '02*, 2002.

[19] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proceedings of the National Conference on Artificial Intelligence*, 1999.

[20] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Information Systems*, 8(2):117–132, March 1997.

[21] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *WebDB Workshop on Databases and the Web*, June 2001.

[22] A. Gupta and I. Mumick, editors. *Materialized Views: Techniques, Implementations and Applications*. The MIT Press, 1999.

[23] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB*, Athens, Greece, 1997.

[24] A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[25] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE*, 2003.

[26] Z. G. Ives, D. Florescu, M. T. Friedman, A. Y. Levy, and D. S. Weld. An adaptive query execution system for data integration. In *SIGMOD '99*, pages 299–310, 1999.

[27] P. Kalnis, W. Ng, B. Ooi, D. Papadias, and K. Tan. An adaptive peer-to-peer network for distributed caching of olap results. In *Proc. of SIGMOD*, 2002.

[28] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *World Wide Web*, pages 150–161, 2001.

[29] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax/, 1999. W3C Recommnedation.

[30] D. B. Lenat and R. Guha. *Building Large Knowledge Bases*. Addison Wesley, Reading Mass., 1990.

[31] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, Bombay, India, 1996.

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

[33] B. McBride. Jena: Implementing the rdf model and syntax specification. http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/, 2001. Hewlett Packard Laboratories.

[34] A. Mendelzon, G. Mihaila, and T. Milo. Querying the world wide web. *International Journal on Digital Libraries*, 1(1):54–67, Apr. 1997.

[35] J. Minker. An overview of cooperative query answering in databases. In *Proceedings of FQAS*, 1998.

[36] P. Mork, S. Gribble, and A. Halevy. Propagating updates in a peer data management system. Unpublished, February 2002.

[37] A. Motro. Accommodating imprecision in database systems: Issues and solutions. *SIGMOD Record*, 19(4):69–74, 1990.

[38] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 190–197, Seattle, WA, USA, 1999. ACM Press.

[39] N. Polyzotis and M. N. Garofalkis. Statistical synopses for graph-structured XML databases. In *SIGMOD '02*, 2002.

[40] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[41] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM '01*, 2001.

[42] M. Rys. Bringing the internet to your database: Using SQLServer 2000 and XML to build loosely-coupled systems. In *ICDE '00*, pages 465–472, 2001.

[43] G. Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[44] S. Soderland. Learning to extract text-based information from the world wide web. In *Knowledge Discovery and Data Mining*, pages 251–254, 1997.

[45] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. of ACM SIGCOMM '01*, 2001.

[46] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 5(1):48–63, 1996.

[47] A. Termier, M.-C. Rousset, and M. Sebag. Treefinder: A first step towards XML data mining. Submitted for publication, 2002.

[48] A. Theobald and G. Weikum. The XXL search engine: Ranked retrieval of XML data using indexes and ontologies. In *SIGMOD '02*, 2002.

[49] H. Treat. Plugging in to XML. *DB2 Magazine*, Winter 1999. Also available at http://www.db2mag.com/winter99/treat.shtml.

[50] K. Wang and H. Liu. Discovering typical structures of documents: A road map approach. In *21st Annual ACM SIGIR Conference*, pages 146–154, 1998.

[51] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. The scale and performance of cooperative web proxy caching. In *SOSP '99*, Kiawah Island, SC, Dec 1999.