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

1 Introduction and Motivation

Online information comes in two flavors: unstructured corpora of text on the 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, management and search 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 is focused 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 an example architecture for a system that crosses the chasm: we describe 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 either do not want to, or simply cannot create a model of their domain. Those who do proceed must invest a significant amount of effort, and they will want some return on their investment.

2. Querying: the difficulties of querying in the S-WORLD are more subtle. In the U-WORLD, a user need not know very 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. On the other hand, in the S-WORLD, a user needs to know the precise schema of the data he wishes to query — otherwise, the query will fail. There is no graceful degradation: if a query is not completely appropriate for the schema, 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. That is, 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).

The points in the above comparison 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* [7]). The problems of creating and sharing structured data are extremely challenging and very 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.

Where is the chasm exactly?

One may wonder where exactly the chasm between structured and unstructured data lies. As a rule of thumb, if posing a query requires that the user know the *domain-specific structure* of the data, then it is an S-WORLD query. Hence, a SQL query on a relational database is an S-WORLD query because the author needs to know the schema. Knowledge bases (and queries over them) are even more elaborate versions of the S-WORLD. An XQuery or XPath posed over XML document structure is also in the S-WORLD[1], but a keyword query on an XML document is in the U-WORLD. A keyword query on a text document is obviously in the U-WORLD as well – while the query author needs to know that the document's structure is a sequence of words, he need not know the domain-specific structure.

1.1.1 Crossing the Chasm

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., [4]), although their coexistence is far from seamless: disparate operations are applied to the different parts.

Our goal is to build tools for the S-WORLD that import that attractive properties of the U-WORLD. We do not expect that managing data in the S-WORLD will ever be as easy as in the U-WORLD, but we claim 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. In particular, we offer the following desiderata for S-WORLD tools:

Authoring: the structure of a content sharing environment can be developed locally and incrementally, matching the requirements and mental model of the content author. Global structure can be developed piecewise across local domains by defining mappings between small subsets of links, and exploiting transitive relationships between these links. Content authoring should be *bottom-up* rather than *top-down*: each author should be free to create content without having to coordinate with other authors.

Querying: while a user is required to learn a *local* structure, he should not be required to learn a global schema that incorporates unfamiliar concepts or models. The user should be able to pose queries from the perspective of the local schema, and the system should map other data into this familiar frame of reference.

Sensitivity to change: Control of the system should be decentralized, both physically and logically. Thus, the system would be robust to changes to both data and schema, which could occur at any place and at any time.

Ease of sharing data: Tools and resources should be available to facilitate mapping between local schemas and sharing of data.

Accuracy of answers: We should retain the S-WORLD properties of providing accurate answers to complex queries, and thus of supporting applications that do not require human intervention.

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. Thus, 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 course at Oxford. DElearning pays for the right to send students to different courses, but charges a 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 as easy as possible for non-technical educators to add in their distance learning courses. Second, it seeks to make tailoring a custom eductational program as easy as possible for a potential customer.

Note that neither the U-WORLD nor the S-WORLD offers technology that can meet DElearning's requirements. U-WORLD technology makes joining DElearning

[1]There are a few subtle exceptions here. For example, one can pose a query that asks for all the elements under the root (or within a fixed distance from the root) without knowing the structure. Similarly, one can ask whether there are any book elements in an XML document without knowing the structure.

easy for educators, they would only need to point DE-learning 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, schedules, and would have to do so across HTML pages constructed using different languages, and different vocabularies. S-WORLD technology sporting a global mediated schema would alleviate these problems, but only at the prohibitive upfront cost of authoring a schema that would cover a large number of universities and departments internationally. □

1.1.2 Roadmap

The paper is organized as follows. In the next section, we describe the REVERE system and describe how it tackles the problems of the structure chasm. Section 3 describes content authoring and local querying support in REVERE. Section 4 describes how REVERE supports data sharing between local domains, and it presents our decentralized system architecture. Section 5 proposes the use of a corpus of structured data and its associated statistics as the basis for tools that facilitate schema creation and mapping. Section 6 puts related work into the perspective of crossing the chasm, and Section 7 concludes.

2 Overview of REVERE

REVERE (Figure 1) is a highly distributed data management system that addresses the problems of the structure chasm on the Web. The goal of REVERE is to build a semantic web 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 supports local migration of HTML data at each site into structured form; provides facilities for establishing mappings between the local schemas of different participants; and presents a query processor that accepts queries over any participant's local schema, and uses the transitive closure of mappings to return all relevant data to the user in his local schema. REVERE is comprised of three components, each designed to import certain attractive properties of the U-WORLD into the S-WORLD. We describe each component in turn below.

Creating structured data

The first hurdle in building any large-scale data sharing system is to structure the existing data. On the top of Figure 1 illustrates the U2S component, which faciliates (and *motivates*) authoring of semantically tagged content *locally* (this component is detailed in Figure 2). 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. To achieve this goal, REVERE tries to replicate the principles behind the Web that we believe made HTML authoring explosively popular:

1. **Ease of use:** we develop an annotation tool that enables users to mark up their data with a given schema, retaining the data in its current location (i.e., inside their HTML pages).

2. **Instant gratification:** on the Web, users enjoy instant gratification as they create HTML files and link them up to others. We mimic that instant gratification by building a set of applications over the structured data, so users can immediately benefit from their annotations.

3. **Deferring integrity constraints:** there are no integrity constraints on the Web. A user can put his phone number on a web page without considering whether it already appears anywhere else (e.g., in an employer's directory). Despite that, users can effectively assess the correctness of the information they find (e.g., by inspecting the URL of the page). Our tool enables users to author any data they want. We shift the burden of enforcing integrity constraints to the applications using the data. Depending on the application, different levels of data cleanliness may be required.

Sharing data

After data has been locally structured, it needs to be shared with other institutions. The second component of the system 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 sense, our PDMS presents a first step in bringing the unconstrained extensibility of the Web into the S-WORLD by distributing data management across multiple nodes and management domains. The principles underlying our PDMS are the following:

1. **Peers:** In a PDMS, peers can serve as data providers, logical mediators, or mere query nodes.

2. **Local mappings:** Data is authored separately on every peer, and 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.

3. **Querying local schemas:** Queries can be formulated using the peer's local schema, and the appropriate translations are made to schemas 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.

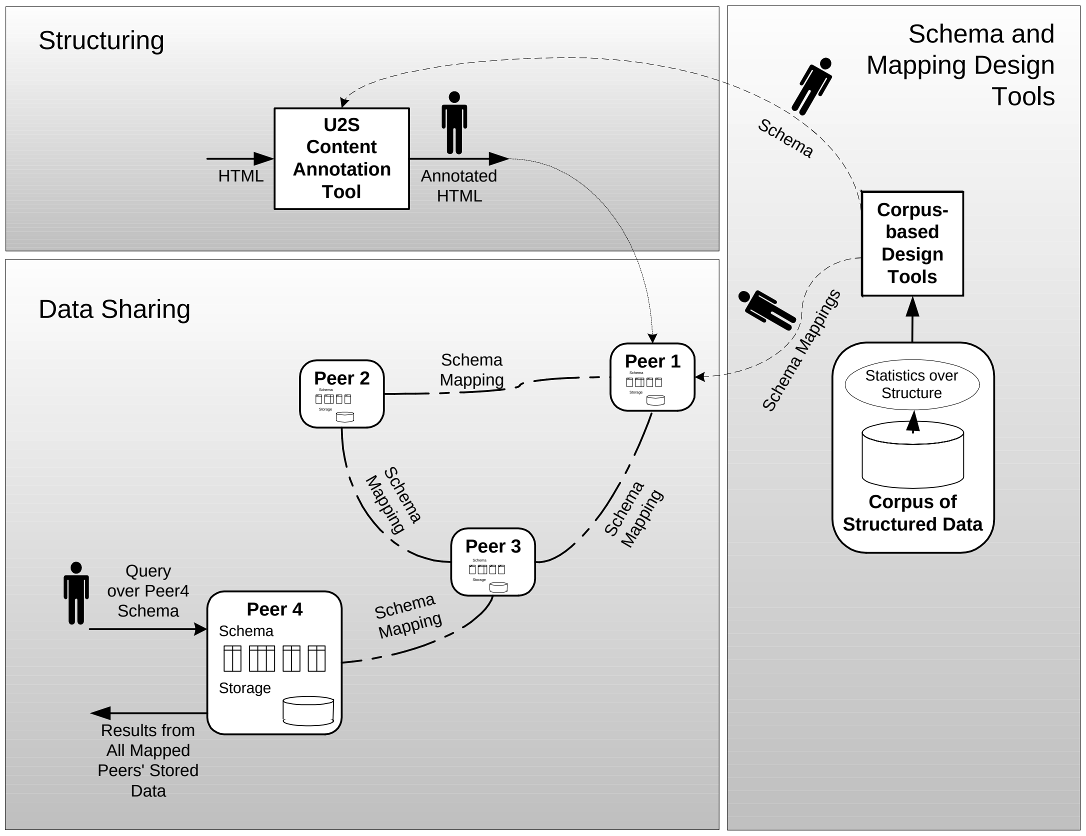


Figure 1: The REVERE system consists of tools for annotating or structuring existing data; a peer-to-peer data sharing environment in which users can pose queries in any of the peers' schemas, and receive answers from all peers; and tools for defining schemas and mappings, which make use of a corpus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

Facilitating authoring and schema mapping

The third component of our system 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.

We note that while we are embarking on an initial corpus effort on our own, we fully expect the creation of the corpus to be a community effort, and the task of devising the most useful statistics to compute from it is a long term research effort.

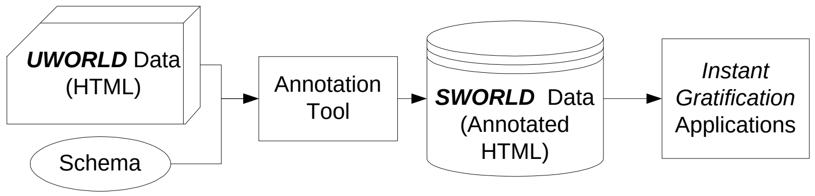
Deploying REVERE for DElearning:

REVERE is ideally suited as a platform for our hypothetical 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 entrance of a new university, or university department into DElearning's network as painless as possible. Since any participating university already has a course web site, all it has to 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 markup 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.

3 The Creation of Structured Data in U2S

In this section, we describe how the U2S module facilitates the authoring of structured content. We consider ease of use, instant gratification applications as an incentive to structure content, and deferred enforcement



UWORLD Data (HTML)
Schema
Annotation Tool
SWORLD Data (Annotated HTML)
Instant Gratification Applications

Figure 2: Architecture of the U2S module. This module faciliates the structuring of data to move it from the U-WORLD to the S-WORLD.

of integrity constraints.

3.1 Ease of Use

To entice a large population of users to structure their data, we must make the process as easy as possible. A huge amount of useful data already exists in personal and organizational web pages, and the active viewing of this data over the web motivates users to keep this information up-to-date. We believe that users should not need to replicate this data in a separate system. Instead, the U2S module facilitates the structuring of such data, by providing an annotation language that allows users to mark-up their data in its current place using a convenient annotation tool. Annotations are embedded in the HTML files but invisible to the browser. This method both ensures backwards compatibility with existing web pages and eliminates inconsistency problems arising from having multiple copies of the same data.

The annotation language: The annotation language is the mechanism that enables adding structure to data in HTML pages. An example of our annotation language is shown in Figure 3. The annotation tags are self-documenting (in the spirit of XML). Two aspects of the language are noteworthy. First, the optional *about* attributes (e.g. of the *course* element) specify unique identifiers (in Uniform Resource Indicator form) for objects. This enables information about an object to be spread in multiple locations and be fused later. Second, for convenience we enable users to tag an entire list or table by specifying a simple regular expression (the **ann:reglist** element) at the top. The "..." refers to the text to be tagged.

Our annotation language is functionally equivalent to basic RDF [45]. We developed our own language instead of using RDF directly for two reasons. First, our language has a simpler and more concise syntax, an important feature for the many users who would prefer to author web pages using a simple text editor. Even more importantly, our language integrates with HTML to allow inline tagging of existing data without disrupting normal browsing, a key requirement for ease of use that is not possible with standard RDF. We expect that a better understanding of the relationship between RDF and XML will emerge shortly (see [59] for a start), and as a result, we will see tools that manipulate the two formalisms uniformly. Currently, we are using the Jena [51] RDF-based storage and querying system in our implementation, because it most closely suits our needs.

Schema in U2S: Users of U2S are required to adhere to one of the schemas provided by the U2S adminis-

```
<html xmlns:ann="http://wash.edu/schema/example">
<ann:course about="U2S://wash.edu/courses/692">
<h1><ann:name>Networking Seminar
    </ann:name></h1>

<p>Office hours for additional assistance:
<ann:instructor>Prof.  John Fitz
    <ann:officeHours>Tue 3-4 pm</ann:officeHours>
</ann:instructor>
<ann:instructor>Prof.  Helen Randolph
    <ann:officeHours>Fri 1-2 pm</ann:officeHours>
</ann:instructor>

<table> <tr><th>2002 Schedule
   <ann:reglist=
      '<tr><ann:event>
         <td><ann:date>...</ann:date>
         <td><ann:topic>...</ann:topic>
      </ann:event></tr>'>

      <tr> <td>Jan 11 <td>Packet loss</tr>
      <tr> <td>Jan 18 <td>TCP theory</tr>
  </ann:reglist>
</table>
</ann:course> </html>
```

Figure 3: Example of annotated HTML. The **ann:** tags provide structured information without disrupting normal HTML browsing.

trator 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 5 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 U2S users are only required to use a set of standardized tag names (and their allowed nesting structure). They are not required to adhere to integrity constraints (that 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 semantics in mind. As we discuss in Section 3.3, we defer the enforcement of integrity constraints.

The annotation tool: Our annotation tool provides a simple GUI interface to enable annotation of existing HTML content. The tool displays a rendered version of the HTML document alongside a tree view of the relevant schema. Users highlight portions of the HTML document, then annotate by choosing a corresponding tag name from the schema. By default, the tool requires users to annotate the document top-down, starting with the top of the schema tree and working down, which assists the user with tag selection and ensures that the resultant structured data conforms to the schema. Advanced users may instead apply annotations in any order and then be warned later about schema violations.

When the user has finished annotating her HTML document, she uses the tool to *publish* her content, as described in the next section.

3.2 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. U2S tries to replicate these conditions for editing S-WORLD data. *Instant gratification* is provided by building a set of applications over U2S 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, "Who's Who", and an annotation-enabled search engine.

When the user *publishes* new content, the annotation tool sends a message to the U2S server, which immediately parses the new document and updates the database. Thus, when the user subsequently executes a U2S-enabled application, she gets the instant gratification of seeing her changes take 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.

For ease of implementation, we currently store the data in a relational database using a simple graph representation. In addition to being updated by *publish* operations, the data is also updated by periodic crawls.

3.3 Treatment of integrity constraints

The third principle underlying the construction of U2S is that on the Web *there are no integrity constraints.* When a user edits her homepage 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 U2S 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 addional 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.

3.4 Example

Returning to our example, 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 U2S annotation tool. Instructors are motivated to annotate (along with perhaps some encouragement from the university!) by the instant gratification they receive from seeing their course added to the U2S-enabled applications. Because the annotated information is part of the standard course page, the information on the page itself 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, U2S also enables some web pages that are currently compiled by hand, such as department-wide course summaries, to be dynamically generated based upon U2S queries in the spririt of systems like Strudel [25].

3.5 U2S and the chasm

The U2S 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 create an environment that entices users to incrementally structure their data, where 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.

4 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. As described in the introduction, this can arise simply because one content developer has a different way of conceptualizing a domain from others; 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 solving the data sharing problem for structured data.

A commonly proposed approach is the one used by data warehousing [16] and data integration [30, 32, 3, 47, 24, 44, 29, 50]: create a common, *mediated* schema that encompasses the concepts to be shared, and define mappings between each source's schema and the mediated schema[2]. 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* 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 his or 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 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.

[2]We observe that the standardized schema of a data warehouse is typically *very* different from the operational data sources' schemas, since it is designed for decision-support queries rather than transactions.

The natural extensibility of a PDMS can best be illustrated by continuing with our running example. In Section 3, we saw how individual university web sites could be annotated with semantic information.

Example 4.1 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 Universita of Rome maps its schema information from Italian to a mediated schema in English, this does not help the University of Trento.

Peer data management techniques are much more appropriate for this task, as shown in Figure 4. 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 peers specifies a mapping to the most convenient other peer(s). Hence, the number of mappings is still 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 [57]

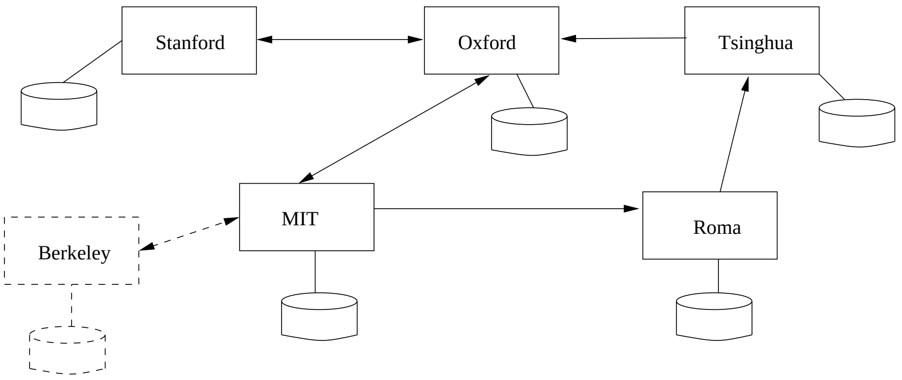


Figure 4: PDMS for our university example. The arrows correspond to schema mappings between peers. No central mediated schema is necessary. As long as the mapping graph is strongly connected, any peer can access data at any other peer by following schema mapping "links".

and Gnutella [23], but also the topic of a significant body of research in distributed systems [18, 70, 10, 67, 62]) 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 in a system called Piazza [31], and we now highlight some aspects of the Piazza system architecture.

4.1 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*[3] 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.

4.1.1 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 [33]): *global-as-view*, in which the mediated schema is defined as a set of queries over the data sources; and *local-as-view*, in which the data sources are defined as views over the mediated schema.

[3]Note that our use of the term "relation" is in a very loose sense, referring to any sort of 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 5: Example peer schemas (in XML DTD form) for Berkeley and MIT.

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 [34] 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" — peer P's relation R is mapped from peer Q's relations S and T — and a given user query may have to be evaluated against the mapping in either the "forwards" or "backwards" direction. This means that our query answering algorithm has aspects of both global-as-view and local-

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

Figure 6: 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.

as-view: it performs query unfolding and query reformulation. 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 5 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 [64] and the annotations used by IBM's XML Extender [69], 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 6, 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.

4.1.2 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 [72] 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" [54]. Updategrams on base data can be combined to create updategrams for views. When a view is updated on a Piazza node, the query optimizer decides which updategrams to use in a cost-based fashion.

4.2 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.

5 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 markup 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 [66] (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 7) from which we will be able to extract extensive

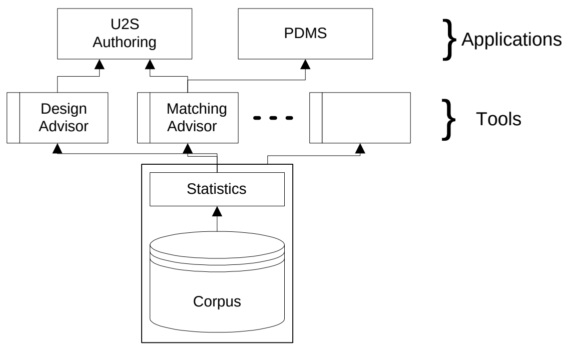


Figure 7: We propose the use of statistical information about structures to alleviate some of the key difficulties of the S-WORLD. The technique is based on collecting corpora (perhaps domain-specific) of structures, and computing a set of statistics on how terms are used in structures. The statistics will be used in a set of general purpose tools that are embedded in various applications.

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.

5.1 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 [46], which attempts to model all common-sense 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 [9], which provides a basic set of operations for manipulating models of data (e.g., schemas, XML DTDs).

5.2 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*.

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?

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 (possiby 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 [71, 28, 60]), and estimate the statistics for other partial structures.

5.3 The Tools

We now describe two tools that will be built using a corpus and associated statistics and used to support components of REVERE. Note that in both cases we envision a tool that interacts with the user to achieve the desired result.

Authoring Structured Data

The first tool, DESIGNADVISOR, assists with the authoring of structured data, and will be used in U2S. 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 otherwise 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 the following:

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.

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 [61] 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 [20] and GLUE [21] 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 finds 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.

5.4 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 U2S) and discovering semantics 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.

6 Related work

While we believe that the problem of crossing the structure chasm has not previously been addressed in a omprehensive 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 [43].

The current approach to building the Semantic Web [7] is based on annotating the data with ontologies. Ontologies, written in an expressive knowledge representation language (e.g., SHOE [37], DAML+OIL [38]), enhance the understanding of the data, and therefore facilitate sharing and integration. Our belief is that different groups inherently need different structures for their data, and that mediation between schemas or ontologies must be a key part of any structured data sharing system. Mediation has received very little attention to date in the Semantic Web community.

In our view, neither of these classes of techniques possesses the level of robustness, accuracy, and scalability required by many applications. Hence, we have focused on an approach that encodes structure within the data, maps across different structures, and queries the structured data.

6.1 Adding and Querying Structure

Structured annotation of HTML pages has been proposed in the "lightweight databases" of [22], but their approach enforced integrity constraints (which we view as too constraining on flexibility), and our system is unique on emphasizing instant gratification and decentralized data sharing across schemas. Other researchers have used the full RDF standard as a way to annotate HTML pages. However, Annotea [39] stores the RDF separately from the document, making the approach less robust to change; CREAM [35] allows the simultaneous creation of new content and associated RDF in the same HTML file, but often requires redundancy between human-readable and machine-readable forms of the data [36]; Haustein and Pleumann [36] store all semantically relevant information in a database and dynamically generate both RDF and human-readable versions from the database, but we believe this is too unwieldy for average users to do in today's web environment. Instead, our annotation language integrates the semantic content with the HTML presentation, allowing easy extraction of structured information from HTML without browser compatibility or redundancy problems.

Ideally, one would not need to annotate HTML at all, and would rely on automated techniques to handle this. In a sense, this is the motivation for information extraction and wrapper induction techniques [42, 65, 49, 5]. These techniques — especially those that support automatic learning of what data to extract (e.g., [42]) — would be a very useful complement to Revere's authoring tool, which currently requires annotations to be made by hand.

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 [11] for a survey of a recent workshop on flexible query answering). Examples include query relaxation [17], cooperative query answering [53], returning approximate and imprecise answers [2, 55], and extensions of SQL or XQuery to support ranked queries over unstructured text fields or elements (e.g., IBM's DB2 Text Extender and [27, 68, 12]). The semantics of these query languages still tend to be heavily 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 [13, 1, 26] or Web hyperlink structure [52] 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).

6.2 Sharing Data from Multiple Schemas

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., [48, 41, 63, 14]). 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 [31] 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 [40] the authors study a variant of the data placement problem, focusing on intelligently caching and reusing queries in an OLAP environment. Recently, [8] described LRM (the local relational model) as a formalism for mediating between different peers in a PDMS, along with an inference procedure for formulas in the language. However, that paper does not describe how LRM's relate in expressive power or efficiency to the more familiar data integration formalisms for mediating between schemas. In contrast, our language is couched as a generalization of schema mediation languages for data integration.

Model management systems [9] are much more general than schema matching tools: they manipulate, map between, and compose schemas (and other structures) in a uniform way. A model management system would be extremely useful as a foundation for our corpus-based tools, as it could provide many of the basic mechanisms for matching schemas and creating schema mappings. Several researchers [61, 6, 20] have even suggested the use of prior mappings to aid in building new ones. Our goals with the corpus include those aspects of model management, but go beyond them: we emphasize obtaining statistical information from the corpus, and we also intend to leverage the corpus not only to help design new mappings, but also as a way of aiding schema designers by suggesting standard structures. We expect that a variety of techniques that have been developed for summarizing, mining, and clustering XML [71, 28, 60] will be useful for computing the statistics associated with the corpus.

Finally, when different data sources are involved, not only are there differences in structure, but there may also be inconsistencies among the data sets or query interfaces. Work has been done on "fusion queries" [73, 56] that try to combine data from multiple sources; on using information-retrieval and probabilistic information to match objects [19, 58], and on doing approximate translations of queries to different sources [15].

7 Conclusion

At first glance, the structure chasm between U-WORLD and the S-WORLD seems all-but-unbridgable 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 re-thinking the design of structured tools for content creation and data sharing. We have presented a detailed path for crossing the chasm by developing the REVERE data system, which facilitate 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 structured corpora 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. Morever, REVERE enables incremental, bottom-up structuring of data rather than requiring the massive, upfront 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.

Acknowledgements

We would like to 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. We thank Phil Bernstein for many thoughtful discussions. Phil Bernstein, Hank Levy, Dan Suciu, and Steve Gribble provided excellent comments on earlier versions of the paper.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Winer. The Lorel query language for semistructured data. In *Proceedings of International Journal on Digital Libraries*, volume 1(1), pages 68–88, April 1997.

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

[3] 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.

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

[5] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with (lixto). In *VLDB '01*, 2001.

[6] 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.

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

[8] 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.

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

[10] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *Proc. Measurement and Modeling of Computer Systems, 2000*, pages 34–43, June 2000.

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

[12] L. J. Brown, M. P. Consens, I. J. Davis, C. R. Palmer, and F. W. Tompa. A structured text ADT for object-relational databases. In *TAPOS*, volume 4(4), pages 227–244, 1998.

[13] P. Buneman, S. B. Davidson, M. F. Fernandez, and D. Suciu. Adding structure to unstructured data. In *ICDT '97*, volume 1186, pages 336–350. Springer, 1997.

[14] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, pages 55–62, 1993.

[15] C.-C. K. Chang and H. Garcia-Molina. Approximate query translation across heterogeneous information sources. In *VLDB '00*, pages 566–577, 2001.

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

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

[18] 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.

[19] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD '98*, pages 201–212, 1998.

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

[21] 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.

[22] S. A. Dobson and V. A. Burrill. Lightweight databases. *Computer Networks and ISDN Systems*, 27(6):1009–1015, 1995.

[23] C. DSS. Gnutella: To the bandwidth barrier and beyond. World Wide Web: www.clip2.com/gnutella.html, November 2000.

[24] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. of PODS*, pages 109–116, Tucson, Arizona., 1997.

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

[26] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *SIGMOD '98*, pages 414–425, 1998.

[27] D. Florescu, I. Manolescu, and D. Kossman. Integrating keyword search into XML query processing. In *Ninth International World Wide Web Conference*, May 2000.

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

[29] M. Friedman and D. Weld. Efficient execution of information gathering plans. In *Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan*, pages 785–791, 1997.

[30] 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.

[31] 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.

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

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

[34] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. Submitted for publication, 2002.

[35] S. Handschuh and S. Staab. Authoring and annotation of web pages in cream. In *World Wide Web*, 2002.

[36] S. Haustein and J. Pleumann. Is participation in the semantic web too difficult? In *Proceedings of the International Semantic Web Conference*, pages 448–453, 2002.

[37] J. Heflin, J. Hendler, and S. Luke. SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078, 1999.

[38] I. Horrocks, F. van Harmelen, and P. Patel-Schneider. DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html, March 2001.

[39] J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared web annotations. In *World Wide Web*, pages 623–632, 2001.

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

[41] R. Krishnamurthy, W. Litwin, and W. Kent. Language features for interoperability of databases with schematic discrepancies. In *Proc. of SIGMOD*, pages 40–49, Denver, Colorado, 1991.

[42] N. Kushmerick, R. Doorenbos, and D. Weld. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

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

[44] E. Lambrecht, S. Kambhampati, and S. Gnanaprakasam. Optimizing recursive information gathering plans. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1204–1211, 1999.

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

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

[47] 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.

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

[49] L. Liu, C. Pu, and W. Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *ICDE '00*, pages 611–621, 2000.

[50] I. Manolescu, D. Florescu, and D. Kossmann. Answering xml queries on heterogeneous data sources. In *Proc. of VLDB*, pages 241–250, 2001.

[51] 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.

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

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

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

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

[56] A. Motro, P. Anokhin, and J. Berlin. Intelligent methods in virtual databases. In *Flexible Query Answering Systems (FQAS) 2000*, pages 580–591, 2000.

[57] Napster. World-wide web: www.napster.com, 2001.

[58] H. Pasula and S. J. Russell. Approximate inference for first-order probabilistic languages. In *ICJCAI '01*, pages 741–748, 2001.

[59] P. Patel-Schneider and J. Simeon. Building the Semantic Web on XML. In *Int'l Semantic Web Conference '02*, pages 147–161, 2002.

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

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

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

[63] M. Rusinkiewicz, A. Sheth, and G. Karabatis. Specifying interdatabase dependencies ina multidatabase environment. *IEEE Computer*, 24:12, 1991.

[64] 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.

[65] A. Sahuguet and F. Azavant. Building light-weight wrappers for legacy web data-sources using W4F. In *VLDB '99*, pages 738–741, 1999.

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

[67] 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.

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

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

[70] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the the Ninth USENIX Security Symposium, Denver, CO*, August 2000.

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

[72] 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.

[73] R. Yerneni, Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Fusion queries over internet databases. In *Proc. of EDBT*, pages 57–71, Valencia, Spain, 1998.