Community Information Management

AnHai Doan[1], Raghu Ramakrishnan[2], Fei Chen[2], Pedro DeRose[1],
Yoonkyong Lee[1], Robert McCann[1], Mayssam Sayyadian[1], Warren Shen[1]
[1] University of Illinois, [2] University of Wisconsin

Abstract

We introduce CIMple*, a joint project between the University of Illinois and the University of Wisconsin.* CIMple *aims to develop a software platform that can be rapidly deployed and customized to manage data-rich online communities. We first describe the envisioned working of such a software platform and our prototype,* DBLife*, which is a community portal being developed for the database research community. We then describe the technical challenges in* CIMple *and our solution approach. Finally, we discuss managing uncertainty and provenance, a crucial task in making our software platform practical.*

1 Introduction

There are many *communities* on the Web, each focusing on a specific set of topics. Examples of communities based on common interests include communities of movie goers, football fans, database researchers, and bioinformaticians. Other common examples include communities with a shared purpose, such as organization intranets and online technical support groups. Community members often want to query, monitor, and discover information about various *entities* and *relationships* in the community. For example, database researchers might be interested in questions such as these:

- Is there any interesting connection between two researchers X and Y (e.g., sharing same advisor)?
- In which course is this paper cited?
- Find all citations of this paper in the past one week on the Web.
- What is new in the past 24 hours in the database research community?

Answering such questions often requires retrieving the raw, largely unstructured data from multiple disparate sources (e.g., home pages, conferences, DBLP, mailing lists), then inferring and monitoring semantic information from the data. Examples of such inference and monitoring include recognizing entity mentions (e.g., "J. Gray", "SIGMOD-04"), deciding if two mentions (e.g., "J. Gray" and "Jim Gray") refer to the same real-world entity, recognizing that a certain relationship (e.g., co-authoring, advising, giving a talk) exists between two entities, detecting that a new entity (e.g., workshop) has appeared, and inferring that a current relationship (e.g., affiliation with a university) has ceased to exist.

The above inference and monitoring tasks are recognized as being challenging [10, 14, 29]. As communities proliferate, the problem of developing effective solutions to support their information needs is becoming increasingly important. We call this problem *community information management*, or *CIM* for short.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

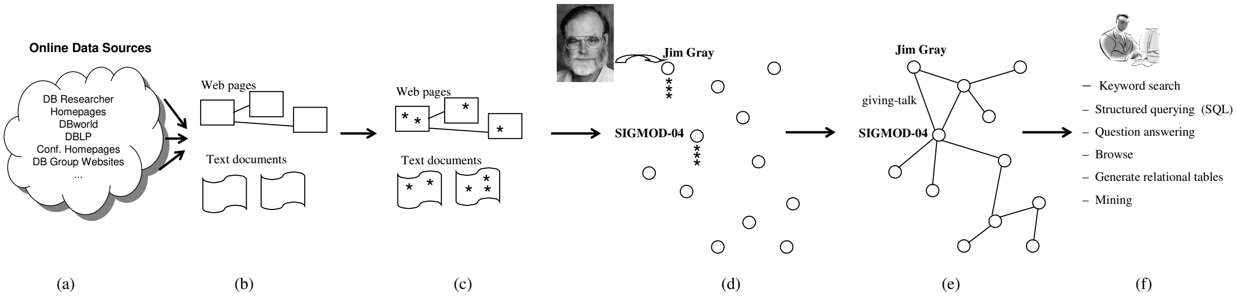

Figure 1: How CIMple infers a semantic entity-relationship graph from the raw unstructured data of the database research community, then leverages the inferred graph to provide a host of user services.

In this paper we describe CIMple[1], a joint project between the University of Illinois and the University of Wisconsin, that addresses the CIM problem. Our goal is to develop a software platform that a specific online community can quickly deploy and customize to effectively manage its data. CIMple can be valuable for communities in a broad range of domains, ranging from scientific data management, government agencies, and enterprise intranets, to those on the World-Wide Web.

We now briefly explain the envisioned working of CIMple, using the community of database researchers as an example (see Figure 1). First, a community expert provides CIMple with a set of relevant data sources (e.g., home pages of database researchers, DBworld mailing list, conference pages, etc.; see Figure 1.a), domain knowledge about entities and relationships of interest, and possibly hints on extracting relevant mentions from the listed data sources. CIMple then crawls the sources at regular intervals to obtain data pages (Figure 1.b), then marks up mentions of relevant entities (denoted by * in Figure 1.c). Examples of mentions include people names (e.g., "J. Gray", "James N. Gray"), conference names, and paper titles. Next, CIMple matches mentions and groups them into entities (Figure 1.d). CIMple then discovers relationships among the entities, in effect transforming the raw data into a semantic entity-relation (ER) data graph (Figure 1.e). CIMple then provides a broad variety of user services over the ER data graph, including browsing, keyword search, structured querying, summarization, and mining. It also maintains and tracks the ER graph over time, as the underlying raw data evolves. Such temporal tracking allows CIMple to provide interesting notification services. Finally, CIMple employs a novel approach called *mass collaboration* to evolve and maintain the inferred ER graph and the provided services. Specifically, it leverages the entire user community, by providing carefully crafted functionalities that are valuable to users, then learning from users' interactions to identify and address a broad range of problems.

Developing the CIMple platform is a long-term goal that we are working toward. As a concrete first step, we are building a prototype community of the kind that CIMple is eventually intended to support. The prototype system, called DBLife, is aimed at the database research community, and the features being developed include monitoring and reporting of interesting events, such as "SIGMOD-06 has posted the list of accepted papers," "researcher X is giving an invited talk at institution Y," and "student S has graduated and moved to department D." To this end, each day we crawl a set of data sources in the database community (e.g., researcher homepages, group pages, conferences, etc.; currently we have collected 1,158 such sources and are adding more). We retrieve Web pages from the sources (on average 11 pages per source, for 20+ MB of data per day), and parse the data to mark up mentions of interesting entities (e.g., researchers, publications, conferences), using hand-crafted rules. We then match the mentions, and group matching mentions into entities. For example, we match roughly 114,400 people mentions per day, and group them into 5,400+ entities. If we see a mention of an entity X on the seminar page of a department Y, then we can infer that X probably is giving a talk at Y. DBLife will be released as an extension of the current DBworld service from the University of Wisconsin.

In the rest of this paper, we briefly describe related work, discuss the technical challenges in developing CIMple and then consider the problem of managing uncertainty and provenance in the CIM context.

[1] The acronym stands for CIM PLatform. The final "e" is an acronym-friendly mistake that we expect our readers to catch.

Related Work: Our work is related to the wealth of research on information extraction and integration (e.g., [1, 22, 16, 4], see [10, 29, 21] for recent tutorials), mention matching, a.k.a. record linkage, entity resolution, fuzzy tuple matching (e.g., [15, 8, 34], see [14] for a brief recent survey), relationship discovery [28], uncertainty and provenance [3, 30, 35, 12, 11, 5]. Many more works exist on these topics than our limited space can list here. Many recent works have also considered the problem of inferring and exploiting semantic information on the World-Wide Web [16], the Semantic Web [33], in personal information management [15], business intelligence (with significant efforts in the AVATAR project [20] and the UIMA framework [17]), text management [19], and most recently, in building data space support platforms (DSSPs), a new unifying data management abstraction for diverse applications [18].

CIMple differs from this body of work in several important ways. First, we focus on *community* settings, where we often have significant domain knowledge about the entities and relationships involved, and hence can address the extraction and integration challenges sufficiently well to enable useful solutions. Second, we seek to build an end-to-end solution to CIM, which requires us to address several important problems that have not received much attention so far. One example is the need to maintain extracted information over time, as the underlying raw data evolves. Third, given a particular community, we want to rapidly deploy and customize our end-to-end solution. Toward this goal, we aim to make our solutions as *declarative* as possible, so that the community builder and users can rapidly enter, debug, and modify domain knowledge that guides the extraction and integration process. Fourth, we must address scalability issues to make our solution practical. To do so, we place a strong emphasis on being able to quickly compile declarative and procedural knowledge (as supplied by a builder and potentially users) into an optimized execution plan, taking cues from optimization technologies in relational contexts. Finally, we add a novel "people" aspect to CIM in particular and information management in general with our use of mass collaboration to improve and maintain data extraction and integration.

2 Technical Challenges

We now discuss four key challenges for CIM: how to extract structure, exploit structure, maintain structure, and provide effective mass collaboration. Our discussion also highlights the sources of uncertainty in CIM contexts. Section 3 then discusses managing uncertainty and the related topic of provenance in more detail.

2.1 Extracting Structure

To deploy CIMple, we propose that the community builder supply a set of seed *Web data sources*, an ER-like *semantic schema* and a set of *extractors*. Since the builder is an expert on the target community, he or she can quickly assemble a set of community data sources to serve as a seed. In the database community, for example, data sources include researcher home pages, DBworld, DBLP, conferences, project pages, etc. CIMple can "bootstrap" from the seed sources to discover other related Web sources.

The builder must also specify an ER-like *semantic schema* whose entities and relationships capture the underlying semantic structure of the domain. For example, in the database domain, entities include person, paper, and conference, and relationships include advise, co-author, and write. A person entity instance has attributes such as name, affiliation, and email. This schema can be specified in its entirety, or in a piecemeal fashion, e.g., some entities and relationships are described first, then some more are added later.

Finally, the builder supplies a set of *extractors*, each of which specifies a way to extract instances of entities or relations of the ER schema from the raw data. Many extraction techniques have been developed (in the areas of information extraction, named entity recognition, wrapper construction, and text segmentation [10, 29, 21]), and many implementations are available commercially, or publicly. The builder thus can create extractors that either directly implement the techniques or adapt off-the-shelf "blackbox" implementations to the target community.

Using the above extraction-related knowledge, CIMple crawls the specified data sources at regular intervals (e.g., daily) to retrieve data pages, then applies extractors to these pages to construct an ER graph of entity and relationship instances that conform to the given ER schema. Conceptually this graph is generated as follows. First, CIMple applies the extractors to the retrieved data pages to extract *mentions* of entities. Example mentions are "J. Gray" and "James N. Gray" for persons and "SIGMOD-04" and "The ACM SIGMOD Conf" for conferences. These mentions are often *ambiguous* in that different ones can refer to the same real-world entity, and conversely the same mention can refer to different real-world entities. CIMple therefore must disambiguate and partition the mentions such that all mentions in a partition refer to the same real-world entity. These entities, denoted as $E_1, \ldots E_n$, form the nodes of the ER graph. In the next step, CIMple applies relation extractors to discover possible relations between the entities. If a relation R exists between two nodes E_i and E_j, then CIMple adds to the ER graph an edge that connects these nodes and corresponds to R.

As described, constructing the ER graph poses two major challenges. First, we must solve the mention disambiguation problem outlined above. Numerous solutions have been proposed to variants of this problem, but they suffer from limited accuracy in the CIM context (as we recently found out when experimenting with the prototype DBLife system), because they typically employ just a single matching technique and thus fail to exploit the varying degree of semantic ambiguity in the data. In recent work [32], we have proposed a solution to this problem, which builds on the observation that different data sources within a community often vary significantly in their level of semantic ambiguity, thus requiring different matching methods.

The second challenge in ER graph construction is that CIMple can apply the mention extractors and mention matchers in many different ways, each of which forms an *execution plan* (analogous in a sense to execution plans in relational databases). These plans often vary significantly in run time and matching accuracy. Hence, finding an optimal or near-optimal execution plan is critical. We are studying the space of such plans, and developing optimizers to find good execution plans, drawing from query optimization insights in relational databases. Our initial work on this topic is described in [32].

2.2 Exploiting Extracted Structure

Once the ER graph has been constructed, CIMple can exploit it to provide many useful services. Services that we are developing or plan to develop include:

- *Keyword search:* Given a keyword query, we return matching data pages, entity instances, relation instances, as well as matching fragments of the ER graphs. We are building on our recent work in keyword search over structured databases [31] to develop this search service.
- *Entity profiling in "Super Homepages":* We create a "super homepage" for each entity E, which displays all information CIMple gathers about that entity, including all mentions of E together with brief explanations. For example, it may say that E's name appeared recently in a conference homepage because E authored a paper in that conference, on a database group homepage because E gave an invited talk, and so on.
- *Notification:* A related functionality is to monitor mentions of an entity, and alert the user when interesting mentions appear. For example, E may want to monitor when a certain paper P will be read by a class or cited in a new paper.
- *ER graph browsing:* Users can browse the ER graph easily, in a manner similar to browsing the citation graph of CiteSeer. For example, starting with a researcher's super homepage, they can follow a paper to a journal in which the paper is published, or follow a community service, e.g., "PC member, SIGMOD-06", to a conference, and so on.
- *Community daily newsletter:* Every morning CIMple will generate an "executive summary" of what interesting events happened in the past 24 hours in the community. For example, the summary may state that X just mentioned on his or her homepage that he or she will serve on SIGMOD-07, or that the list of papers accepted to ICDE-06 has just been posted on DBworld.
- *Structured querying:* We are studying how to formulate SQL-like structured queries over the extracted ER graph, what the syntax and semantics of the results should be, and how to execute such queries efficiently. As

a first step, [24] describes our recent work on interactive SQL querying of Web data generated from structured templates (e.g., Amazon.com and DBLP pages).

• *Temporal keyword search and structured querying:* We will develop temporal query capabilities. For example, the user can pose a query over only the data crawled in the past two weeks, or ask CIMple to rank answers in chronological order. Note that this is a key advantage of CIMple over general search engines. Such search engines cover the *entire* World-Wide Web and hence cannot archive it, e.g., on a daily basis, in order to provide effective temporal querying.

• *Service modules for one-click capabilities in context:* We will allow community builders to use the full range of search, query, alert and reporting capabilities to associate stored versions (of searches, queries, etc.) with different entity and relationship types. When a page displays entities or relationship instances of these types, the associated capabilities are then made available to users. Thus, even casual users can ask sophisticated queries through a single click in context.

2.3 Maintaining Extracted Structures

After creating an ER graph and associated services for a community, we must monitor and adjust the ER graph as the underlying data sources evolve. Sources on the Web are often highly dynamic [23, 9]. Consequently, maintaining the extracted ER graph is a labor intensive undertaking, and developing techniques to reduce the maintenance cost is critical.

The first maintenance challenge is how to *efficiently update* the extracted ER graph. We can simply re-crawl the sources at regular intervals (e.g., daily), then apply the methods discussed previously to rebuild the ER graph from scratch. Indeed, we apply this solution in the current DBLife prototype. However, this solution has two major limitations (as confirmed painfully by our experience with the prototype). First, rebuilding the ER graph from scratch is quite time intensive. We must extract afresh all entity mentions, match the mentions, and rediscover the various relations. All of these tasks are time consuming [10, 29, 21]. Thus, for highly dynamic communities (e.g., auction or finance) that require updating the ER graph every few hours, this approach will probably not scale. Second, if we rebuild the ER graph from scratch at every re-crawl, we loose the temporal dimension associated with entities and relationships. Consider an entity E that we have inferred from the raw data on Day 1. If we re-crawl and rebuild the ER graph again on Day 2, which of the newly created entities correspond to E? Establishing the correspondence is crucial for tracking E over time, and for answering temporal queries. To remove these limitations, we are currently developing a solution to *incrementally* update the ER graph.

The second challenge that we must address is how to detect and repair broken extractors. Consider for example a price extractor that always returns as a price the third number in the first italics line of a page. This extractor will break if data pages change the formatting rule used to display prices. Given the central role extractors play in CIMple it is important that we detect and repair broken extractors, or adjust to them in some way when repair has not been carried out. We have developed an initial solution called Maveric for detecting broken extractors [23], and are developing a solution that attempts to repair a broken extractor or makes repair suggestions to the builder.

2.4 Mass Collaboration

We have described how a builder can use CIMple to quickly develop and deploy a first-cut data portal for a community. Next, we propose a set of novel techniques that leverages *community users*—instead of the *builder*—to help refine the extraction and integration logic used by CIMple. We consider three specific techniques: personalized data spaces, reputation incentives, and "payment" schemes.

• *Personalized data spaces:* To illustrate the idea of leveraging personalized data spaces for the common good, consider the Internet Movie Database (IMDB, at *imdb.com*). When a new movie is added to IMDB, within a few days it receives a score (say 7.6 out of 10), averaged over scores given by thousands of movie goers. How can IMDB convince so many people to score the movie? It turns out that the vast majority of scores come from *registered users*. These users maintain *private* movie collections in their IMDB accounts, i.e., personalized versions of the public IMDB movie collection. Many users then score the movies in their collections only so that they can search later, e.g., "list all action movies in my collection that I gave a score of 7 or higher", using the IMDB search engine. IMDB, however, can aggregate these private scores to provide public scores for movies. Our idea is to similarly allow users to personalize and manage private versions of the public data space in their CIMple accounts, then learn from the private actions (in a way that protects individual privacy and has their consent) to improve the public data space. We will consider a range of "personalization" actions that users can carry out in their private CIMple accounts, design simple interfaces to facilitate them, then develop techniques to leverage personalization activities to improve the public data portal.

• *Reputation incentives:* There were cases where researchers corrected mistakes in their DBLP homepages by contacting the DBLP owners, because they felt that these homepages form an important part of their public image. This is similar to the experience of many community portals; e.g., at QUIQ [26], people answered hundreds (in some cases, thousands) of technical support questions posed by other users solely to gain greater visibility in the community through a variety of mechanisms. We therefore plan to design CIMple such that users will have sufficiently strong "reputation incentives" to correct mistakes in the publicly viewable (extracted and integrated, and therefore possibly incorrect) information that they believe may adversely affect them. This raises several challenges that we plan to examine: First, how can we authenticate a user vis a vis the corresponding person entity? Second, if different users edit related data (e.g., co-authors edit the same paper), their changes may conflict. How should we reconcile the changes? Finally, how can we provide sufficiently strong incentives?

• *Payment schemes:* For certain CIM services, we can make users "pay" for using them, by answering relatively simple questions. We then leverage the answers to improve the services. For example, suppose that CIMple has compiled a query processing bibliography B, and that employing learning techniques, it has found a paper P that may be relevant to B. Then when a user U wants to access B, CIMple may show U the paper P and ask "Is P about query processing?" Once the user has answered, he or she is allowed to access B. If a sufficient number of users answer yes, then CIMple may decide to include P in bibliography B, thereby expanding B. To realize the above idea, we must address several challenges—the most important is merging multiple, often noisy, user answers into a single answer. Users may disagree and there may not even be a single correct answer. We discuss these challenges and preliminary solutions in [25, 27, 26]. The QUIQ experience [27, 26] and experiments on small user communities (10-130 users) [25] suggest the potential of the mass collaboration approach.

3 Managing Uncertainty and Provenance

We now discuss uncertainty and provenance problems in CIMple, and sketch our solutions. We highlight in particular the ideas of making CIM services interactive and leveraging mass collaboration to reduce uncertainty.

3.1 Uncertainty

All major CIM steps—extraction and integration, data evolution, and mass collaboration—generate uncertainty. Techniques for extracting mentions, tagging them (e.g., as people names, conferences, etc.), and matching them are well-known to be imperfect. In addition, the evolution of the data often invalidates the assumptions made by extractors, causing errors, thus adding yet another source of uncertainty. Finally, mass collaboration generates uncertainty since people do not contribute in a perfect fashion, and malicious or ignorant users often provide incorrect feedback.

To handle such uncertainties, we consider three directions: (a) understand the nature of the uncertainty involved, (b) model and reason using uncertain data, and (c) reduce uncertainty.

Understanding Uncertainty: In CIM contexts, we have encountered primarily two types of uncertainty: *confidence score* and *multi-value*, though additional types are certainly possible. The first type arises when an extractor (or a mention matcher) operates on domain knowledge or heuristics that are typically (though not always) true. In such cases the CIM module can make qualified predictions, such as "the mention Madison in a text document U is a last name with confidence 0.8" or "mentions Madison and J. Madison match with confidence 0.9." Most current extractors generate this type of uncertainty.

The second type of uncertainty, multi-value, arises when an extractor operates on *correct, but underspecified* domain knowledge. For instance, given a text document U and the knowledge "there is a publication year in U and it is of numeric type" (e.g., supplied by the user [24]), an extractor may extract the set of all numeric values from U, say $\{23, 4, 2001, 1998, 135\}$, and specify that publication year takes a value in this set. As another example, if values are organized into a domain hierarchy (e.g., time might be specified as days, weeks, and years), we might not always know a value at the finest granularity. Thus, we might know that a sale occurred in 2006, but not know the specific week or day. Multi-value uncertainty, specifically the form arising from domain hierarchies, is referred to as *imprecision* in [6].

Reasoning with Uncertainty: We are developing methods to provide services such as keyword search and SQL querying over extracted uncertain data. We have studied confidence-score uncertainty in the context of keyword search over multiple disparate and heterogeneous structured data sets. Given a user query Q, the goal is to return a ranked list of answers, where each answer "glues" together a set of data fragments taken from the same or different data sets [31]. Since the structured data sets are heterogeneous, matching mentions (e.g., "D. Smith" vs. "David Smith") and matching schema elements (e.g., pname vs. researcher-name) can be predicted only with some confidence score. Our solution is to incorporate such scores directly into the overall score of each answer for the keyword query. This work suggests that IR-style ranking can be a very natural tool for handling multiple types of uncertainties.

For the second type of uncertainty (i.e., multi-value), we have developed a solution to provide SQL querying over extracted data with multi-value uncertainty [24]. Our solution provides a *superset semantics*, i.e., it always produces a superset of the correct results. A key idea underlying the solution is that it is *interactive*: it allows the user to quickly pose SQL queries, obtain initial results, then iterate to get increasingly better results. In each iteration, the system asks the user 1-2 relatively simple questions, designed to solicit structural information to *reduce* the uncertainty associated with multiple values. It then leverages the user answers to refine the query results.

In joint work with the Avatar project at IBM [20], we have also developed a principled approach to defining semantics for OLAP queries over imprecise data [6] based on *allocation* of imprecise facts, and are developing efficient implementation techniques for allocation [7].

Reducing Uncertainty: We reduce uncertainty via two mechanisms: user interaction and mass collaboration. For many CIM services (e.g., keyword search, SQL querying), we consider how to make them interactive, so that the service can learn from the user, and provide increasingly better results (in a sense this strategy can be viewed as a variant of relevance user feedback, see [31, 24]). Our preliminary work on this topic in the context of keyword search is described in [31], and an interactive solution for SQL querying over multi-valued data is described in [24].

In CIMple, we also apply mass collaboration to reduce uncertainty, a solution that, to the best of our knowledge, has not been considered in the context of data extraction or integration. As discussed in Section 2.4, we will develop this solution in two steps. First, we allow each user to take extracted data provided by automatic solutions as a starting point, then manually correct the data or interact with CIMple to improve the data. Second, we develop solutions to learn from what each user is doing, and apply the results to help other users.

3.2 Provenance

In CIM contexts, we found that provenance serves two goals. First, it helps the user understand and reduce uncertainty. Second, it provides a way for the user to express certain information needs using provenance-related criteria (e.g., find only answers that originate from sources X and Y).

Toward the first goal, we are designing CIMple to provide a *justification* for any answer it produces in response to a user query. The justification lists all data pages that contribute to the answer, and all operations (extraction, mention matching, etc.) that have been invoked along the path from the raw data to the answer. Thus, in effect we provide a "derivation (a.k.a., provenance) tree" that explains how the answer was produced. The user can follow the derivation tree to the original data pages for further verification. We also plan to develop a "what-if" facility. When examining the derivation tree, the user can ask hypothetical questions, such as "What if I state that these two mentions do not match?" The idea is that CIMple will show how the answers look if the proposed assumption holds. Such a facility would help evaluate the robustness of the original answers, thereby increasing user confidence in those answers. We have developed justification mechanisms previously for schema matching [13] and logic program inference [2]. We are building upon this work, as well as results on managing provenance (e.g., [3, 12, 5]) to develop justification and "what-if" mechanisms for CIMple. Toward the second goal of serving provenance-related information needs, we will adopt results from current work on languages for querying provenance (e.g., [3, 5]) to the CIM context, on an "as-needed" basis.

4 Concluding Remarks

We have introduced the CIMple project that develops a software platform to extract and integrate information for online communities. In general, information extraction is taking on an increasingly larger role in how we seek to organize and analyze text corpora. By its nature, extracted information has associated uncertainty, and the CIM setting offers many novel opportunities and challenges for dealing with this uncertainty, as we have tried to illustrate in this brief overview of CIMple.

References

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from Web pages. In *SIGMOD-03*.

[2] T. Arora, R. Ramakrishnan, W. Roth, P. Seshadri, and D. Srivastava. Explaining program execution in deductive systems. In *Int. Conf. on Deductive and Object-Oriented Databases*, 1993.

[3] O. Benjelloun, A. Das Sarma, C. Hayworth, and J. Widom. An introduction to ULDBs and the Trio system. *IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases*, 29(1), 2006.

[4] L. Bertossi, J. Chomicki, P. Godfrey, P. Kolaitis, A. Thomo, and C. Zuzarte. Exchange, integration, and consistency of data: Report on the ARISE/NISR workshop. *SIGMOD Record*, 34(3):87–90, 2005.

[5] P. Buneman, S. Khanna, K. Tajima, and W. Tan. Archiving scientific data. *ACM Trans. on Database Systems*, 29:2–42, 2004.

[6] D. Burdick, P. Deshpande, T. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. In *VLDB-05*.

[7] D. Burdick, P. Deshpande, T. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Efficient allocation algorithms for OLAP over imprecise data. *IBM Almaden Tech. Report*, 2006.

[8] S. Chaudhuri, V. Ganti, and R. Motwani. Robust identification of fuzzy duplicates. In *ICDE-05*.

[9] W. Cohen. Some practical observations on integration of Web information. In *WebDB-99*.

[10] W. Cohen. Information extraction. *Tutorial, www.cs.cmu.edu/∼wcohen/ie-survey.ppt*, 2003.

[11] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB-04*.

[12] N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *VLDB-05*.

[13] R. Dhamankar, Y. Lee, A. Doan, P. Domingos, and A. Halevy. iMAP: Learning complex matches between database schemas. In *SIGMOD-04*.

[14] A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine, Special Issue on Semantic Integration*, Spring 2005.

[15] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD-05*.

[16] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in KnowItAll. In *WWW-04*.

[17] D. Ferrucci and A. Lally. Building an example application with the unstructured information management architecture. *IBM Systems Journal 43, No. 3, 455-475*, 2004.

[18] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems (invited paper). In *PODS-06*.

[19] P. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. To search or to crawl? Towards a query optimizer for text-centric tasks. In *SIGMOD-06*.

[20] R. Krishnamurthy, S. Raghavan, J. Thathachar, S. Vaithyanathan, and H. Zhu. AVATAR information extraction system. *IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases*, 29(1), 2006.

[21] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira. A brief survey of Web data extraction tools. *SIGMOD Record*, 31(2), 2002.

[22] I. Mansuri and S. Sarawagi. A system for integrating unstructured data into relational databases. In *ICDE-06*.

[23] R. McCann, B. AlShelbi, Q. Le, H. Nguyen, L. Vu, and A. Doan. Mapping maintenance for data integration systems. In *VLDB-05*.

[24] R. McCann, P. DeRose, A. Doan, and R. Ramakrishnan. SLIC: On-the-fly extraction and integration of Web data. *UW-Madison Tech. Report TR-1558, available as http://anhai.cs.uiuc.edu/public/slic-tr06.pdf*, 2006.

[25] R. McCann, A. Doan, A. Kramnik, and V. Varadarajan. Building data integration systems via mass collaboration. In *WebDB-03*.

[26] R. Ramakrishnan. Mass collaboration and data mining, 2001. Keynote address, KDD-01, www.cs.wisc.edu/ raghu/Kddrev.ppt.

[27] R. Ramakrishnan, A. Baptist, V. Ercegovac, M. Hanselman, N. Kabra, A. Marathe, and U. Shaft. Mass collaboration: A case study. In *IDEAS-04*.

[28] D. Roth and W. Yih. Probabilistic reasoning for entity and relation recognition. In *Int. Conf. on Computational Linguistics (COLING)*, 2002.

[29] S. Sarawagi. Graphical models for structure extraction and information integration. *Keynote talk and tutorial at ICDM-05, www.it.iitb.ac.in/ sunita/talks/graphical2.ppt*, 2005.

[30] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *ICDE-06*.

[31] M. Sayyadian, A. Doan, and L. Gravano. Keyword search across heterogeneous relational databases. In *UIUC Tech. Report 03-2006-02, available as http://anhai.cs.uiuc.edu/public/kite-tr06.pdf*, 2006.

[32] W. Shen, P. DeRose, L. Vu, A. Doan, and R. Ramakrishnan. Source-aware entity matching: A compositional approach. *UW-Madison Tech. Report TR-1559, available as http://anhai.cs.uiuc.edu/public/soccer-tr06.pdf*, 2006.

[33] Frank van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.

[34] M. Weis and F. Naumann. Dogmatix tracks down duplicates in XML. In *SIGMOD-05*.

[35] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR-05*.