Semantic Integration Research in the Database Community: A Brief Survey

AnHai Doan
University of Illinois
anhai@cs.uiuc.edu

Alon Y. Halevy
University of Washington
alon@cs.washington.edu

Semantic integration has been a long-standing challenge for the database community. It has received steady attention over the past two decades, and has now become a prominent area of database research.

In this article, we first review database applications that require semantic integration, and discuss the difficulties underlying the integration process. We then describe recent progress and identify open research issues. We will focus in particular on schema matching, a topic that has received much attention in the database community, but will also discuss data matching (e.g., tuple deduplication), and open issues beyond the match discovery context (e.g., reasoning with matches, match verification and repair, and reconciling inconsistent data values). For previous surveys of database research on semantic integration, see (Rahm & Bernstein 2001; Ouksel & Seth 1999; Batini, Lenzerini, & Navathe 1986).

Applications of Semantic Integration

The key commonalities underlying database applications that require semantic integration are that they use structured representations (e.g., relational schemas and XML DTDs) to encode the data, and that they employ more than one representation. As such, the applications must resolve heterogeneities with respect to the schemas and their data, either to enable their manipulation (e.g., merging the schemas or computing the differences (Batini, Lenzerini, & Navathe 1986; Bernstein 2003)) or to enable the translation of data and queries across the schemas. Many such applications have arisen over time and have been studied actively by the database community.

One of the earliest such applications is *schema integration*: merging a set of given schemas into a single global schema (Batini, Lenzerini, & Navathe 1986; Elmagarmid & Pu 1990; Seth & Larson 1990; Parent & Spaccapietra 1998; Pottinger & Bernstein 2003). This problem has been studied since the early 1980s. It arises in building a database system that comprises several distinct databases, and in designing the schema of a database from the local schemas supplied by several user groups. The integration process requires establishing semantic correspondences— matches—between the component schemas, and then using the matches to merge schema elements (Pottinger & Bernstein 2003; Batini, Lenzerini, & Navathe 1986).

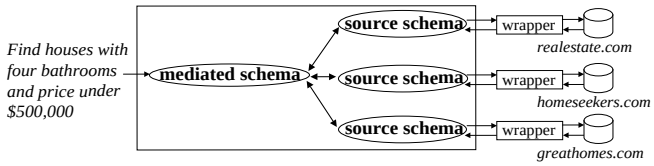

Figure 1: A data integration system in the real estate domain. Such a system uses the semantic correspondences between the mediated schema and the source schemas (denoted with double-head arrows in the figure) to reformulate user queries.

As databases become widely used, there is a growing need to *translate data* between multiple databases. This problem arises when organizations consolidate their databases and hence must transfer data from old databases to the new ones. It forms a critical step in *data warehousing* and *data mining*, two important research and commercial areas since the early 1990s. In these applications, data coming from multiple sources must be transformed to data conforming to a single target schema, to enable further data analysis (Miller, Haas, & Hernandez 2000; Rahm & Bernstein 2001).

In the recent years, the explosive growth of information online has given rise to even more application classes that require semantic integration. One application class builds *data integration systems* (e.g., (Garcia-Molina *et al.* 1997; Levy, Rajaraman, & Ordille 1996; Ives *et al.* 1999; Lambrecht, Kambhampati, & Gnanaprakasam 1999; Friedman & Weld 1997; Knoblock *et al.* 1998)). Such a system provides users with a uniform query interface (called *mediated schema*) to a multitude of data sources, thus freeing them from manually querying each individual source.

Figure 1 illustrates a data integration system that helps users find houses on the real-estate market. Given a user query over the mediated schema, the system uses a set of semantic matches between the mediated schema

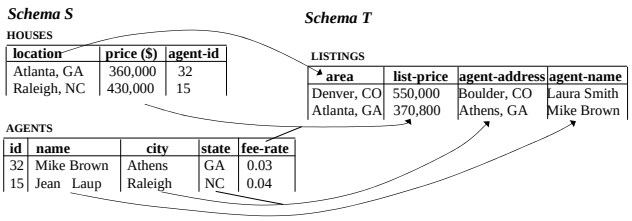

Figure 2: The schemas of two relational databases S and T on house listing, and the semantic correspondences between them. Database S consists of two tables: HOUSES and AGENTS; database T consists of the single table LISTINGS.

and the local schemas of the data sources to translate it into queries over the source schemas. Next, it executes the queries using wrapper programs attached to the sources (e.g., (Kushmerick, Weld, & Doorenbos 1997)), then combines and returns the results to the user. A critical problem in building a data integration system, therefore, is to supply the semantic matches. Since in practice data sources often contain duplicate items (e.g., the same house listing) (Hernandez & Stolfo 1995; Bilenko & Mooney 2003; Tejada, Knoblock, & Minton 2002), another important problem is to detect and eliminate duplicate data tuples from the answers returned by the sources, before presenting the final answers to the user query.

Another important application class is *peer data management*, which is a natural extension of data integration (Aberer 2003). A peer data management system does away with the notion of mediated schema and allows peers (i.e., participating data sources) to query and retrieve data directly from each other. Such querying and data retrieval require the creation of semantic correspondences among the peers.

Recently there has also been considerable attention on *model management*, which creates tools for easily manipulating models of data (e.g., data representations, website structures, and ER diagrams). Here semantic integration plays a central role, as matching and merging models form core operations in model management algebras (Bernstein 2003; Rahm & Bernstein 2001).

The data sharing applications described above arise in numerous current real-world domains. They also play an important role in emerging domains such as e-commerce, bioinformatics, and ubiquitous computing. Some recent developments should dramatically increase the need for and the deployment of applications that require semantic integration. The Internet has brought together millions of data sources and makes possible data sharing among them. The widespread adoption of XML as a standard syntax to share data has further streamlined and eased the data sharing process. The growth of the Semantic Web will further fuel data sharing applications and underscore the key role that semantic integration plays in their deployment.

Challenges of Semantic Integration

Despite its pervasiveness and importance, semantic integration remains an extremely difficult problem. Consider, for example, the challenges that arise during a schema matching process, which finds semantic correspondences (called *matches*) between database schemas. For example, given the two relational databases on house listing in Figure 2, the process finds matches such as "location in schema S matches area in schema T" and "name matches agent-name".

At the core, matching two database schemas S and T requires deciding if any two elements s of S and t of T *match*, that is, if they refer to the same real-world concept. This problem is challenging for several fundamental reasons:

- The semantics of the involved elements can be inferred from only a few information sources, typically the creators of data, documentation, and associated schema and data. Extracting semantics information from data creators and documentation is often extremely cumbersome. Frequently, the data creators have long moved, retired, or forgotten about the data. Documentation tends to be sketchy, incorrect, and outdated. In many settings such as when building data integration systems over remote Web sources, data creators and documentation are simply not accessible.
- Hence schema elements are typically matched based on clues in the *schema* and *data*. Examples of such clues include element names, types, data values, schema structures, and integrity constraints. However, these clues are often unreliable. For example, two elements that share the same name (e.g., area) can refer to different real-world entities (the location and square-feet area of the house). The reverse problem also often holds: two elements with different names (e.g., area and location) can refer to the same real-world entity (the location of the house).
- Schema and data clues are also often incomplete. For example, the name contact-agent only suggests that the element is related to the agent. It does not provide sufficient information to determine the exact nature of the relationship (e.g., whether the element is about the agent's phone number or her name).
- To decide that element s of schema S matches element t of schema T, one must typically examine *all* other elements of T to make sure that there is no other element that matches s better than t. This *global* nature of matching adds substantial cost to the matching process.
- To make matters worse, matching is often *subjective*, depending on the application. One application may decide that house-style matches house-description, another application may decide that it does not. Hence, the user must often be involved in the matching process. Sometimes, the input of a single user may be

considered too subjective, and then a whole committee must be assembled to decide what the correct mapping is (Clifton, Housman, & Rosenthal 1997).

Because of the above challenges, the manual creation of semantic matches has long been known to be extremely laborious and error-prone. For example, a recent project at the GTE telecommunications company sought to integrate 40 databases that have a total of 27,000 elements (i.e., attributes of relational tables) (Li & Clifton 2000). The project planners estimated that, without the original developers of the databases, just finding and documenting the matches among the elements would take more than 12 person years.

The problem of matching data tuples also faces similar challenges. In general, the high cost of manually matching schemas and data has spurred numerous solutions that seek to *automate the matching process*. Because the users must often be in the loop, most of these solutions have been semi-automatic. Research on these solutions dates back to the early 80s, and has picked up significant steam in the past decade, due to the need to manage the astronomical volume of distributed and heterogeneous data at enterprises and on the Web. In the next two sections we briefly review this research on schema and data matching.

Schema Matching

We discuss the accummulated progress in schema matching with respect to matching techniques, architectures of matching solutions, and types of semantic matches.

Matching Techniques

A wealth of techniques has been developed to semi-automatically find semantic matches. The techniques fall roughly into two groups: rule-based and learning-based solutions (though several techniques that leverage ideas from the fields of information retrieval and information theory have also been developed (Clifton, Housman, & Rosenthal 1997; Kang & Naughton 2003)).

Rule-Based Solutions: Many of the early as well as current matching solutions employ hand-crafted rules to match schemas (Milo & Zohar 1998; Palopoli, Sacca, & Ursino 1998; Castano & Antonellis 1999; Mitra, Wiederhold, & Jannink ; Madhavan, Bernstein, & Rahm 2001; Melnik, Molina-Garcia, & Rahm 2002).

In general, hand-crafted rules exploit *schema information* such as element names, data types, structures, number of subelements, and integrity constraints. A broad variety of rules have been considered. For example, the TranScm system (Milo & Zohar 1998) employs rules such as "two elements match if they have the same name (allowing synonyms) and the same number of subelements". The DIKE system (Palopoli, Sacca, & Ursino 1998; Palopoli *et al.* 1999; Palopoli, Terracina, & Ursino 2000) computes the similarity between two schema elements based on the similarity of the characteristics of the elements and the similarity of related elements. The ARTEMIS and the related MOMIS (Castano & Antonellis 1999; Bergamaschi *et al.* 2001) system compute the similarity of schema elements as a weighted sum of the similarities of name, data type, and substructure. The CUPID system (Madhavan, Bernstein, & Rahm 2001) employs rules that categorize elements based on names, data types, and domains. Rules therefore tend to be domain-independent, but can be tailored to fit a certain domain. Domain-specific rules can also be crafted.

Rule-based techniques provide several benefits. First, they are relatively inexpensive and do not require training as in learning-based techniques. Second, they typically operate only on schemas (not on data instances), and hence are fairly fast. Third, they can work very well in certain types of applications and for domain representations that are amenable to rules (Noy & Musen 2000).

Finally, rules can provide a quick and concise method to capture valuable user knowledge about the domain. For example, the user can write regular expressions that encode times or phone numbers, or quickly compile a collection of county names or zip codes that help recognize those types of entities. As another example, in the domain of academic course listing, the user can write the following rule: "use regular expressions to recognize elements about times, then match the first time element with start-time and the second element with end-time". Learning techniques, as we discuss shortly, would have difficulties being applied to these scenarios. They either cannot learn the above rules, or can do so only with abundant training data or with the right representations for training examples.

The main drawback of rule-based techniques is that they cannot exploit data instances effectively, even though the instances can encode a wealth of information (e.g., value format, distribution, frequently occurring words in the attribute values, and so on) that would greatly aid the matching process. In many cases effective matching rules are simply too difficult to hand craft. For example, it is not clear how to hand craft rules that distinguish between "movie description" and "user comments on the movies", both being long textual paragraphs. In contrast, learning methods such as Naive Bayes can easily construct "probabilistic rules" that distinguish the two with high accuracy, based on the frequency of words in the paragraphs.

Another drawback is that rule-based methods cannot exploit previous matching efforts to assist in the current ones. Thus, in a sense, systems that rely solely on rule-based techniques have difficulties learning from the past, to improve over time. The above reasons have motivated the development of learning based matching solutions.

Learning-Based Solutions: Many such solutions have been proposed in the past decade, e.g., (Li, Clifton, & Liu 2000; Clifton, Housman, & Rosenthal 1997;

Berlin & Motro 2001; 2002; Doan, Domingos, & Halevy 2001; Dhamankar *et al.* 2004; Embley, Jackman, & Xu 2001; Neumann *et al.* 2002). The solutions have considered a variety of learning techniques and exploited both schema and data information. For example, the SemInt system (Li, Clifton, & Liu 2000) uses a neural-network learning approach. It matches schema elements based on attribute specifications (e.g, data types, scale, the existence of constraints) and statistics of data content (e.g., maximum, minimum, average, and variance). The LSD system (Doan, Domingos, & Halevy 2001) employs Naive Bayes over data instances, and develops a novel learning solution to exploit the hierarchical nature of XML data. The iMAP system (Dhamankar *et al.* 2004) (and also the ILA and HICAL systems developed in the AI community (Perkowitz & Etzioni 1995; Ryutaro, Hideaki, & Shinichi 2001)) matches the schemas of two sources by analyzing the description of objects that are found in both sources. The Autoplex and Automatch systems (Berlin & Motro 2001; 2002) use a Naive Bayes learning approach that exploits data instances to match elements.

In the past five years, there is also a growing realization that schema- and data-related evidence in two schemas being matched often is inadequate for the matching process. Hence, several works have advocated learning from the *external evidence* beyond the two current schemas. Several types of external evidence have been considered. Some recent works advocate exploiting *past matches* (Doan, Domingos, & Halevy 2001; Do & Rahm 2002; Berlin & Motro 2002; Rahm & Bernstein 2001; Embley, Jackman, & Xu 2001; Bernstein *et al.* 2004). The key idea is that a matching tool must be able to learn from the past matches, to predict successfully matches for subsequent, unseen matching scenarios.

The work (Madhavan *et al.* 2005) goes further and describes how to exploit a *corpus of schemas and matches* in the domain. This scenario arises, for example, when we try to exploit the schemas of numerous real-estate sources on the Web, to help in matching two specific real-estate source schemas. In a related direction, the works (He & Chang 2003; Wu *et al.* 2004) describe settings where one must *match multiple schemas* all at once. Here the knowledge gleaned from each matching pair can help match other pairs, as a result we can obtain better accuracy than just matching a pair in isolation. The work (McCann *et al.* 2003) discusses how to learn from a *corpus of users* to assist schema matching in data integration contexts. The basic idea is to ask the *users* of a data integration system to "pay" for using it by answering relatively simple questions, then use those answers to further build the system, including matching the schemas of the data sources in the system. This way, an enormous burden of schema matching is lifted from the system builder and spread "thinly" over a mass of users.

Architecture of Matching Solutions

The complementary nature of rule- and learner-based techniques suggest that an effective matching solution should employ both – each on the types of information that it can effectively exploit. To this end, several recent works (Bernstein *et al.* 2004; Do & Rahm 2002; Doan, Domingos, & Halevy 2001; Embley, Jackman, & Xu 2001; Rahm, Do, & Massmann 2004; Dhamankar *et al.* 2004) have described a system architecture that employs multiple modules called *matchers*, each of which exploits well a certain type of information to predict matches. The system then combines the predictions of the matchers to arrive at a final prediction for matches. Each matcher can employ one or a set of matching techniques as described earlier (e.g., hand-crafted rules, learning methods, IR-based ones). Combining the predictions of matchers can be manually specified (Do & Rahm 2002; Bernstein *et al.* 2004) or automated to some extent using learning techniques (Doan, Domingos, & Halevy 2001).

Besides being able to exploit multiple types of information, the multi-matcher architecture has the advantage of being highly modular and can be easily customized to a new application domain. It is also extensible in that new, more efficient matchers could be easily added when they become available. A recent work (Dhamankar *et al.* 2004) also shows that the above solution architecture can be extended successfully to handle complex matches.

An important current research direction is to evaluate the above multi-matcher architecture in real-world settings. The works (Bernstein *et al.* 2004; Rahm, Do, & Massmann 2004) make some initial steps in this direction. A related direction appears to be a shift away from developing complex, isolated, and monolithic matching systems, towards creating robust and widely useful matcher operators and developing techniques to quickly and efficiently combine the operators for a particular matching task.

Incorporating Domain Constraints: It was recognized early on that domain integrity constraints and heuristics provide valuable information for matching purposes. Hence, almost all matching solutions exploit some forms of this type of knowledge.

Most works exploit integrity constraints in matching schema elements *locally*. For example, many works match two elements if they participate in similar constraints. The main problem with this scheme is that it cannot exploit "global" constraints and heuristics that relate the matching of *multiple* elements (e.g., "at most one element matches house-address"). To address this problem, several recent works (Melnik, Molina-Garcia, & Rahm 2002; Madhavan, Bernstein, & Rahm 2001; Doan, Domingos, & Halevy 2001; Doan *et al.* 2003b) have advocated moving the handling of constraints to *after* the matchers. This way, the constraint handling framework can exploit "global" constraints and

is highly extensible to new types of constraints.

While integrity constraints constitute *domain-specific* information (e.g., house-id is a key for house listings), heuristic knowledge makes *general* statements about how the matching of elements relate to each other. A well-known example of a heuristic is "two nodes match if their neighbors also match", variations of which have been exploited in many systems (e.g., (Milo & Zohar 1998; Madhavan, Bernstein, & Rahm 2001; Melnik, Molina-Garcia, & Rahm 2002; Noy & Musen 2001)). The common scheme is to *iteratively* change the matching of a node based on those of its neighbors. The iteration is carried out one or twice, or until some convergence criterion is reached.

Related Work in Knowledge-Intensive Domains: Schema matching requires making *multiple interrelated inferences*, by combining a *broad variety* of relatively *shallow* knowledge types. In recent years, several other problems that fit the above description have also been studied in the AI community. Notable problems are information extraction (e.g., (Freitag 1998)), solving crossword puzzles (Keim *et al.* 1999), and identifying phrase structure in NLP (Punyakanok & Roth 2000). What is remarkable about these studies is that they tend to develop similar solution architectures which combine the prediction of multiple independent modules and optionally handle domain constraints on top of the modules. These solution architectures have been shown empirically to work well. It will be interesting to see if such studies converge in a definitive blueprint architecture for making multiple inferences in knowledge-intensive domains.

Types of Semantic Matches

Most schema matching solutions have focused on finding 1-1 matches such as "location = address". However, relationships between real-world schemas involve many *complex matches*, such as "name = concat(first-name,last-name)" and "listed-price = price * (1 + tax-rate)". Hence, the development of techniques to semi-automatically construct complex matches is crucial to any practical mapping effort.

Creating complex matches is fundamentally harder than 1-1 matches for the following reason. While the number of candidate 1-1 matches between a pair of schemas is bounded (by the product of the sizes of the two schemas), the number of candidate complex matches is not. There are an unbounded number of functions for combining attributes in a schema, and each one of these could be a candidate match. Hence, in addition to the inherent difficulties in generating a match to start with, the problem is exacerbated by having to examine an unbounded number of match candidates.

There have been only a few works on complex matching. Milo and Zohar (1998)) hard-code complex matches into rules. The rules are systematically tried on the given schema pair, and when such a rule fires, the system returns the complex match encoded in the rule. Several recent works have developed more general techniques to find complex matches. They rely on a domain ontology (Xu & Embley 2003), use a combination of search and learning techniques (Dhamankar *et al.* 2004; Doan *et al.* 2003b), or employ mining techniques (He, Chang, & Han 2004). Xu and Embley (2003), for example, considers finding complex matches between two schemas by first mapping them into a domain ontology, then constructing the matches based on the relationships inherent in that ontology. The iMAP system reformulates schema matching as a *search* in an often very large or infinite match space. To search effectively, it employs a set of searchers, each discovering specific types of complex matches.

Perhaps the key observation gleaned so far from the above few works is that we really need *domain knowledge* (and lots of it!) to perform accurate complex matching. Such knowledge is crucial in guiding the process of searching for likely complex match candidates (in a vast or often infinite candidate space), in pruning incorrect candidates early (to maintain an acceptable runtime), and in evaluating candidates.

Another important observation is that the correct complex match is often not the top-ranked match, but somewhere in the top few matches predicted. Since finding a complex match requires gluing together so many different components (e.g., the elements involved, the operations, etc.), perhaps this is inevitable and inherent to any complex matching solution. This underscores the importance of generating explanations for the matches, and building effective match design environment, so that humans can effectively examine the top ranked matches to select the correct ones.

Data Matching

Besides schema matching, the problem of data matching (e.g., deciding if two different relational tuples from two sources refer to the same real-world entity) is also becoming increasingly crucial. Popular examples of data matching include matching citations of research papers, authors, and institutions. As another example, consider again the databases in Figure 2. Suppose we have created the mappings, and have used them to transfer the house listings from database S and another database U (not shown in the figure) to those of database T. Databases S and U may contain many duplicate house listings. Hence in the next step we would like to detect and merge such duplicates, to store and reason with the data at database T.

The above tuple matching problem has received much attention in the database, AI, KDD, and WWW communities, under the names merge/purge, tuple deduplication, entity matching (or consolidation), and object matching.

Research on tuple matching has roughly paralleled that of schema matching, but slightly lagged behind in certain aspects. Just like in schema matching, a

variety of techniques for tuple matching has been developed, including both rule-based and learning-based approaches Early solutions employ manually specified rules (Hernandez & Stolfo 1995), while many subsequent ones learn matching rules from training data (Tejada, Knoblock, & Minton 2002; Bilenko & Mooney 2003; Sarawagi & Bhamidipaty 2002). Several solutions focus on efficient techniques to match strings (Monge & Elkan 1996; Gravano *et al.* 2003). Others also address techniques to scale up to very large number of tuples (McCallum, Nigam, & Ungar 2000; Cohen & Richman 2002). Several recent methods have also heavily used information retrieval (Cohen 1998; Ananthakrishna, Chaudhuri, & Ganti 2002) and information-theoretic (Andritsos, Miller, & Tsaparas 2004) techniques.

Recently, there has also been some efforts to exploit *external* information to aid tuple matching. The external information can come from past matching efforts and domain data (e.g., see the paper by Martin Michalowski *et. al.* in this issue). In addition, many works have considered the settings where there are many tuples to be matched, and examined how information can be moved across different matching pairs, to improve matching accuracy (Parag & Domingos 2004; Bhattacharya & Getoor 2004).

At the moment, a definitive solution architecture for tuple matching has not yet emerged, though the work (Doan *et al.* 2003a) proposes a multi-module architecture reminiscent to the multi-matcher architecture of schema matching. Indeed, given that tuple matching and schema matching both try to infer semantic relationships on the basis of limited data, the two problems appear quite related, and techniques developed in one area could be transferred to the other. This implication is significant because so far these two active research areas have been developed quite independently of each other.

Finally, we note that some recent works in the database community have gone beyond the problem of matching tuples, into matching data fragments in text and semi-structured data (Dong *et al.* 2004; Fang *et al.* 2004), a topic that has also been receiving increasing attention in the AI community (e.g., see the paper by Xin Li *et. al.* in this special issue).

Open Research Directions

Matching schemas and data usually constitute only the first step in the semantic integration process. We now discuss open issues related to this first step, as well as to some subsequent important steps that have received little attention.

User Interaction: In many cases, matching tools must interact with the user to arrive at final correct matches. We consider efficient user interaction one of the most important open problems for schema matching. Any practical matching tool must handle this problem, and anecdotal evidence abounds that deployed matching tools are quickly being abandoned because they irritate users with too many questions. Several recent works have only touched on this problem (e.g., (Yan *et al.* 2001)). An important challenge here is to minimize user interaction by asking for absolutely necessary feedback, but maximizing the impact of feedback. Another challenge is to generate effective explanations of matches (Dhamankar *et al.* 2004).

Formal Foundations: In parallel with efforts to build pratical matching systems, several recent works have developed formal semantics of matching and attempted to explain formally what matching tools are doing (e.g., (Larson, Navathe, & Elmasri 1989; Biskup & Convent 1986; Madhavan *et al.* 2002; Sheth & Kashyap 1992; Kashyap & Sheth 1996)). Formalizing the notion of semantic similarity has also received some attention (Ryutaro, Hideaki, & Shinichi 2001; Lin 1998; Manning & Schütze 1999). Nevertheless, this topic remains underdeveloped. It should deserve more attention, because such formalizations are important for the purposes of evaluating, comparing, and further developing matching solutions.

Industrial Strength Schema Matching: Can current matching techniques be truly useful in real-world settings? Are we solving the right schema matching problems? Partly to answer these questions, several recent works seek to evaluate the applicability of schema matching techniques in the real world. The work (Bernstein *et al.* 2004) attempts to build an industrial strength schema matching environment, while the work (Rahm, Do, & Massmann 2004) focuses on scaling up matching techniques, specifically on matching large XML schemas, which are common in practice. The works (Seligman *et al.* 2002; Rosenthal, Seligman, & Renner 2004) examine the difficulties of real-world schema matching, and suggest changes in data management practice that can facilitate the matching process. These efforts should help us understand better the applicability of current research and suggest future directions.

Mapping Maintenance: In dynamic environments, sources often undergo changes in their schemas and data. Hence, it is important to *evolve* the discovered semantic mappings. A related problem is to detect changes at autonomous data sources (e.g., those on the Internet), verify if the mappings are still correct, and repair them if necessary. Despite the importance of this problem, it has received relatively little attention (Kushmerick 2000; Lerman, Minton, & Knoblock 2003; Velegrakis, Miller, & Popa 2003).

Reasoning with Imprecise Matches on a Large Scale: A large-scale data integration or peer-to-peer systems will inevitably involve thousands or hundreds of thousands of semantic mappings. At this scale, it will be impossible for human to verify and maintain all of them, to ensure the correctness of the system. How can

we use systems where parts of the mappings always remain unverified and potentially incorrect? In a related problem, it is unrealistic to expect that some day our matching tools will generate only perfect mappings. If we can generate only reasonably good mappings, on a large scale, are they good for any purpose? Note that these problems will be crucial at any large scale data integration and sharing scenario, such as the Semantic Web.

Schema Integration: In schema integration, once matches among a set of schemas have been identified, the next step uses the matches to merge the schemas into a global schema (Batini, Lenzerini, & Navathe 1986). A closely related research topic is model management (Bernstein 2003; Rahm & Bernstein 2001). As described earlier, model management creates tools for easily manipulating models of data (e.g., data representations, website structures, and ER diagrams). Here matches are used in higher-level operations, such as merging schemas and computing difference of schemas. Several recent works have discussed how to carry out such operations (Pottinger & Bernstein 2003), but they remain very difficult tasks.

Data Translation: In these applications we often must elaborate matches into mappings, to enable the translation of queries and data across schemas. (Note that here we follow the terminologies of (Rahm & Bernstein 2001) and distinguish between *match* and *mapping*, as described above.) In Figure 2, for example, suppose the two databases that conform to schemas S and T both store house listings and are managed by two different real-estate companies.

Now suppose the companies have decided to merge. To cut costs, they eliminate database S by transferring all house listings from S to database T. Such data transfer is not possible without knowing the exact *semantic mappings* between the relational schemas of the databases, which specify how to create data for T from data in S. An example mapping, shown in SQL notation, is

```
list-price = SELECT price * (1 + fee-rate)
             FROM   HOUSES, AGENTS
             WHERE  agent-id = id
```

In general, a variety of approaches have been used to specify semantic mappings (e.g., SQL, XQuery, GAV, LAV, GLAV (Lenzerini 2002)).

Elaborating a semantic match, such as "list-price = price * (1 + fee-rate)", that has been discovered by a matching tool, into the above mapping is a difficult problem, and has been studied by (Yan *et al.* 2001), which developed the Clio system. How to combine mapping discovery systems such as Clio with schema matching systems to build a unified and effective solution for finding semantic mappings is an open research problem.

Peer-to-Peer Data Management: An emerging important application class is *peer data management*, which is a natural extension of data integration (Aberer 2003). A peer data management system does away with the notion of mediated schema and allows peers (i.e., participating data sources) to query and retrieve data directly from each other. Such querying and data retrieval require the creation of semantic mappings among the peers. Peer data management also raises novel semantic integration problems such as composing mappings among peers to enable the transfer of data and queries between two peers with no direct mappings, and dealing with loss of semantics during the composition process (Etzioni *et al.* 2003).

Concluding Remarks

We have briefly surveyed the broad range of semantic integration research in the database community. The paper (and the special issue in general) demonstrates that this research effort is quite related to those in the AI community. It is also becoming clear that semantic integration lies at the heart of many database and AI problems, and that addressing it will require solutions that blend database and AI techniques. Developing such solutions can be greatly facilitated with even more effective collaboration between the various communities in the future.

Acknowledgment: We thank Natasha Noy for invaluable comments on the earlier drafts of this paper.

References

Aberer, K. 2003. Special issue on peer to peer data management. *SIGMOD Record* 32(3).

Ananthakrishna, R.; Chaudhuri, S.; and Ganti, V. 2002. Eliminating fuzzy duplicates in data warehouses. In *Proc. of 28th Int. Conf. on Very Large Databases.*

Andritsos, P.; Miller, R. J.; and Tsaparas, P. 2004. Information-theoretic tools for mining database structure from large data sets. In *Proc. of the ACM SIGMOD Conf.*

Batini, C.; Lenzerini, M.; and Navathe, S. 1986. A comparative analysis of methodologies for database schema integration. *ACM Computing Survey* 18(4):323–364.

Bergamaschi, S.; Castano, S.; Vincini, M.; and Beneventano, D. 2001. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering* 36(3):215–249.

Berlin, J., and Motro, A. 2001. Autoplex: Automated discovery of content for virtual databases. In *Proceedings of the Conf. on Cooperative Information Systems (CoopIS).*

Berlin, J., and Motro, A. 2002. Database schema matching using machine learning with feature selection. In *Proceedings of the Conf. on Advanced Information Systems Engineering (CAiSE).*

Bernstein, P. A.; Melnik, S.; Petropoulos, M.; and Quix, C. 2004. Industrial-strength schema matching. *SIGMOD Record, Special Issue in Semantic Integration.* To appear.

Bernstein, P. 2003. Applying model management to classical meta data problems. In *Proceedings of the Conf. on Innovative Database Research (CIDR).*

Bhattacharya, I., and Getoor, L. 2004. Iterative record linkage for cleaning and integration. In *Proc. of the 9th*

ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

Bilenko, M., and Mooney, R. 2003. Adaptive duplicate detection using learnable string similarity measures. In *KDD Conf.*

Biskup, J., and Convent, B. 1986. A formal view integration method. In *Proceedings of the ACM Conf. on Management of Data (SIGMOD).*

Castano, S., and Antonellis, V. D. 1999. A schema analysis and reconciliation tool environment. In *Proceedings of the Int. Database Engineering and Applications Symposium (IDEAS).*

Clifton, C.; Housman, E.; and Rosenthal, A. 1997. Experience with a combined approach to attribute-matching across heterogeneous databases. In *Proc. of the IFIP Working Conference on Data Semantics (DS-7).*

Cohen, W., and Richman, J. 2002. Learning to match and cluster entity names. In *Proc. of 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.*

Cohen, W. 1998. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Procceedings of SIGMOD-98.*

Dhamankar, R.; Lee, Y.; Doan, A.; Halevy, A.; and Domingos, P. 2004. iMAP: Discovering complex matches between database schemas. In *Proc. of the ACM SIGMOD Conf. (SIGMOD).*

Do, H., and Rahm, E. 2002. Coma: A system for flexible combination of schema matching approaches. In *Proceedings of the 28th Conf. on Very Large Databases (VLDB).*

Doan, A.; Lu, Y.; Lee, Y.; and Han, J. 2003a. Object matching for data integration: A profile-based approach. In *IEEE Intelligent Systems, Special Issue on Information Integration*, volume 18.

Doan, A.; Madhavan, J.; Dhamankar, R.; Domingos, P.; and Halevy, A. 2003b. Learning to match ontologies on the Semantic Web. *VLDB Journal* 12:303–319.

Doan, A.; Domingos, P.; and Halevy, A. 2001. Reconciling schemas of disparate data sources: A machine learning approach. In *Proceedings of the ACM SIGMOD Conference.*

Dong, X.; Halevy, A.; Nemes, E.; Sigurdsson, S.; and Domingos, P. 2004. Semex: Toward on-the-fly personal information integration. In *Proc. of the VLDB IIWeb Workshop.*

Elmagarmid, A., and Pu, C. 1990. Guest editors' introduction to the special issue on heterogeneous databases. *ACM Computing Survey* 22(3):175–178.

Embley, D.; Jackman, D.; and Xu, L. 2001. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Proceedings of the WIIW Workshop.*

Etzioni, O.; Halevy, A.; Doan, A.; Ives, Z.; Madhavan, J.; McDowell, L.; and Tatarinov, I. 2003. Crossing the structure chasm. In *Conf. on Innovative Database Research.*

Fang, H.; Sinha, R.; Wu, W.; Doan, A.; and Zhai, C. 2004. Entity retrieval over structured data. Technical Report UIUC-CS-2414, Dept. of Computer Science, Univ. of Illinois.

Freitag, D. 1998. Machine learning for information extraction in informal domains. *Ph.D. Thesis.* Dept. of Computer Science, Carnegie Mellon University.

Friedman, M., and Weld, D. 1997. Efficiently executing information-gathering plans. In *Proc. of the Int. Joint Conf. of AI (IJCAI).*

Garcia-Molina, H.; Papakonstantinou, Y.; Quass, D.; Rajaraman, A.; Sagiv, Y.; Ullman, J.; and Widom, J. 1997. The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Inf. Systems* 8(2).

Gravano, L.; Ipeirotis, P.; Koudas, N.; and Srivastava, D. 2003. Text join for data cleansing and integration in an rdbms. In *Proc. of 19th Int. Conf. on Data Engineering.*

He, B., and Chang, K. 2003. Statistical schema matching across web query interfaces. In *Proc. of the ACM SIGMOD Conf. (SIGMOD).*

He, B.; Chang, K. C. C.; and Han, J. 2004. Discovering complex matchings across Web query interfaces: A correlation mining approach. In *Proc. of the ACM SIGKDD Conf. (KDD).*

Hernandez, M., and Stolfo, S. 1995. The merge/purge problem for large databases. In *SIGMOD Conference*, 127–138.

Ives, Z.; Florescu, D.; Friedman, M.; Levy, A.; and Weld, D. 1999. An adaptive query execution system for data integration. In *Proc. of SIGMOD.*

Kang, J., and Naughton, J. 2003. On schema matching with opaque column names and data values. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD-03).*

Kashyap, V., and Sheth, A. 1996. Semantic and schematic similarities between database objects: A context-based approach. *The VLDB Journal* 5(4):276–304.

Keim, G.; Shazeer, N.; Littman, M.; Agarwal, S.; Cheves, C.; Fitzgerald, J.; Grosland, J.; Jiang, F.; Pollard, S.; and Weinmeister, K. 1999. PROVERB: The probabilistic cruciverbalist. In *Proc. of the 6th National Conf. on Artificial Intelligence (AAAI-99)*, 710–717.

Knoblock, C.; Minton, S.; Ambite, J.; Ashish, N.; Modi, P.; Muslea, I.; Philpot, A.; and Tejada, S. 1998. Modeling web sources for information integration. In *Proc. of the National Conference on Artificial Intelligence (AAAI).*

Kushmerick, N.; Weld, D.; and Doorenbos, R. 1997. Wrapper Induction for Information Extraction. In *Proc. of IJCAI-97.*

Kushmerick, N. 2000. Wrapper verification. *World Wide Web Journal* 3(2):79–94.

Lambrecht, E.; Kambhampati, S.; and Gnanaprakasam, S. 1999. Optimizing recursive information gathering plans. In *Proc. of the Int. Joint Conf. on AI (IJCAI).*

Larson, J. A.; Navathe, S. B.; and Elmasri, R. 1989. A theory of attribute equivalence in database with application to schema integration. *IEEE Transaction on Software Engineering* 15(4):449–463.

Lenzerini, M. 2002. Data integration; a theoretical perspective. In *Proc. of PODS-02.*

Lerman, K.; Minton, S.; and Knoblock, C. A. 2003. Wrapper maintenance: a machine learning approach. *Journal of Artificial Intelligence Research.* To appear.

Levy, A. Y.; Rajaraman, A.; and Ordille, J. 1996. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB.*

Li, W., and Clifton, C. 2000. SEMINT: A tool for identifying attribute correspondence in heterogeneous databases using neural networks. *Data and Knowledge Engineering* 33:49–84.

Li, W.; Clifton, C.; and Liu, S. 2000. Database integration using neural network: implementation and experience. *Knowledge and Information Systems* 2(1):73–96.

Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Madhavan, J.; Bernstein, P.; and Rahm, E. 2001. Generic schema matching with Cupid. In *Proceedings of the International Conference on Very Large Databases (VLDB)*.

Madhavan, J.; Halevy, A.; Domingos, P.; and Bernstein, P. 2002. Representing and reasoning about mappings between domain models. In *Proceedings of the National AI Conference (AAAI-02)*.

Madhavan, J.; Bernstein, P.; Doan, A.; and Halevy, A. 2005. Corpus-based schema matching. In *Proc. of the 18th IEEE Int. Conf. on Data Engineering (ICDE)*.

Manning, C., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, US: The MIT Press.

McCallum, A.; Nigam, K.; and Ungar, L. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.

McCann, R.; Doan, A.; Kramnik, A.; and Varadarajan, V. 2003. Building data integration systems via mass collaboration. In *Proc. of the SIGMOD-03 Workshop on the Web and Databases (WebDB-03)*.

Melnik, S.; Molina-Garcia, H.; and Rahm, E. 2002. Similarity flooding: a versatile graph matching algorithm. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

Miller, R.; Haas, L.; and Hernandez, M. 2000. Schema mapping as query discovery. In *Proc. of VLDB*.

Milo, T., and Zohar, S. 1998. Using schema matching to simplify heterogeneous data translation. In *Proceedings of the International Conference on Very Large Databases (VLDB)*.

Mitra, P.; Wiederhold, G.; and Jannink, J. Semi-automatic integration of knowledge sources. In *Proceedings of Fusion'99*.

Monge, A., and Elkan, C. 1996. The field matching problem: Algorithms and applications. In *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*.

Neumann, F.; Ho, C.; Tian, X.; Haas, L.; and Meggido, N. 2002. Attribute classification using feature analysis. In *Proceedings of the Int. Conf. on Data Engineering (ICDE)*.

Noy, N., and Musen, M. 2000. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Noy, N., and Musen, M. 2001. Anchor-PROMPT: Using non-local context for semantic Matching. In *Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*.

Ouksel, A., and Seth, A. P. 1999. Special issue on semantic interoperability in global information systems. *SIGMOD Record* 28(1).

Palopoli, L.; Sacca, D.; Terracina, G.; and Ursino, D. 1999. A unififed graph-based framework for deriving nominal interscheme properties, type conflicts, and object cluster similarities. In *Proceedings of the Conf. on Cooperative Information Systems (CoopIS)*.

Palopoli, L.; Sacca, D.; and Ursino, D. 1998. Semi-automatic, semantic discovery of properties from database schemes. In *Proc. of the Int. Database Engineering and Applications Symposium (IDEAS-98)*, 244–253.

Palopoli, L.; Terracina, G.; and Ursino, D. 2000. The system DIKE: towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *Proceedings of the ADBIS-DASFAA Conf.*

Parag, and Domingos, P. 2004. Multi-relational record linkage. In *Proc. of the KDD Workshop on Multi-relational Data Mining*.

Parent, C., and Spaccapietra, S. 1998. Issues and approaches of database integration. *Communications of the ACM* 41(5):166–178.

Perkowitz, M., and Etzioni, O. 1995. Category translation: Learning to understand information on the Internet. In *Proc. of Int. Joint Conf. on AI (IJCAI)*.

Pottinger, R. A., and Bernstein, P. A. 2003. Merging models based on given correspondences. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*.

Punyakanok, V., and Roth, D. 2000. The use of classifiers in sequential inference. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS-00)*.

Rahm, E., and Bernstein, P. 2001. On matching schemas automatically. *VLDB Journal* 10(4).

Rahm, E.; Do, H.; and Massmann, S. 2004. Matching large XML schemas. *SIGMOD Record, Special Issue in Semantic Integration*. To appear.

Rosenthal, A.; Seligman, L.; and Renner, S. 2004. From semantic integration to semantics management: case studies and a way forward. *SIGMOD Record, Special Issue in Semantic Integration*. To appear.

Ryutaro, I.; Hideaki, T.; and Shinichi, H. 2001. Rule induction for concept hierarchy alignment. In *Proceedings of the 2nd Workshop on Ontology Learning at the 17th Int. Joint Conf. on AI (IJCAI)*.

Sarawagi, S., and Bhamidipaty, A. 2002. Interactive deduplication using active learning. In *Proc. of 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.

Seligman, L.; Rosenthal, A.; Lehner, P.; and Smith, A. 2002. Data integration: Where does the time go? *IEEE Data Engineering Bulletin*.

Seth, A., and Larson, J. 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Survey* 22(3):183–236.

Sheth, A. P., and Kashyap, V. 1992. So far (schematically) yet so near (semantically). In *Proc. of the IFIP WG 2.6 Database Semantics Conf. on Interoperable Database Systems*.

Tejada, S.; Knoblock, C.; and Minton, S. 2002. Learning domain-independent string transformation weights for high

accuracy object identification. In *Proc. of the 8th SIGKDD Int. Conf. (KDD-2002)*.

Velegrakis, Y.; Miller, R. J.; and Popa, L. 2003. Mapping adaptation under evolving schemas. In *Proc. of the Conf. on Very Large Databases (VLDB)*.

Wu, W.; Yu, C.; Doan, A.; and Meng, W. 2004. An interactive clustering-based approach to integrating source query interfaces on the Deep Web. In *Proc. of the ACM SIGMOD Conf.*

Xu, L., and Embley, D. 2003. Using domain ontologies to discover direct and indirect matches for schema elements. In *Proc. of the Semantic Integration Workshop at ISWC-03, http://smi.stanford.edu/si2003*.

Yan, L.; Miller, R.; Haas, L.; and Fagin, R. 2001. Data driven understanding and refinement of schema mappings. In *Proceedings of the ACM SIGMOD*.