Building Data Integration Systems: A Mass Collaboration Approach

AnHai Doan   Robert McCann
{anhai,rlmccann}@cs.uiuc.edu
Department of Computer Science
University of Illinois, Urbana-Champaign, IL 61801, USA

Abstract

Building data integration systems today is largely done by hand, in a very labor intensive and error prone process. In this paper, we describe a conceptually new solution to this problem: that of mass collaboration. The basic idea is to think about a data integration system as having a finite set of parameters whose values must be set. To build such a system, the system administrators can construct and deploy a system "shell", then ask the users to help the system "automatically converge" to the correct parameter values. This way, the enourmous burden of system developments is lifted from the administrators and spread "thinly" over a multitude of users. We discuss the challenges to this approach and propose solutions. We then describe our current effort in applying this approach to the problem of schema matching in the context of data integration.

Introduction

The rapid growth of distributed data on the Internet and at enterprises has generated much interests in building data integration systems. Such systems provide a *uniform* query interface to a multitude of data sources, thereby freeing the user from the tedious task of interacting and combining data from the individual sources. Figure 1 shows a data integration system over several sources that list books for sell. Given a user query that is formulated in the query interface (also called the *mediated schema*), the system uses a set of *semantic mappings* to translate the query into queries over *source schemas*, then executes the queries and combines the data returned from the sources, to produce the desired answers to the user.

Numerous research activities have been conducted on data integration, both in the AI and database communities (Garcia-Molina *et al.* 1997; Levy, Rajaraman, & Ordille 1996; Haas *et al.* 1997; Yerneni, Papakonstantinou, & Garcia-Molina 1998; Ives *et al.* 1999; Kwok & Weld 1996; Friedman & Weld 1997; Lambrecht, Kambhampati, & Gnanaprakasam 1999; Duschka & Genesereth 1997; Knoblock *et al.* 1998; Arens, Hsu, & Knoblock 1996; Chen *et al.* 2000; Avnur & Hellerstein 2000). Much progress has been made in terms of developing conceptual and algorithmic frameworks; query optimization; constructing semi-automatic tools for schema matching, wrapper construction, and object matching; and fielding data integration

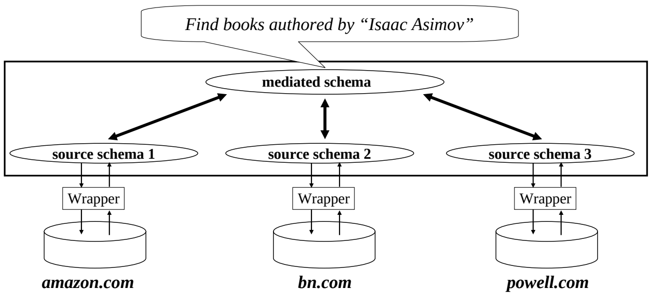

Figure 1: A data integration system in the book domain

systems on the Internet.

Despite substantial progress, however, today building data integration systems is still largely done by hand, in an extremely labor intensive and error prone process. The advent of languages and mediums for creating and exchanging semi-structured data, such as XML, OWL, and the Semantic Web, will further accelerate the needs for data integration systems and exacerbate the above problem. Thus, it has now become critical to develop techniques that allow the efficient construction and maintenance of data integration systems.

In this paper we describe the MOBS (Mass Collaboration to Build Systems) approach to efficiently building data integration systems. The basic idea underlying our approach is to treat a data integration system as having a finite set of parameters whose values must be set. The system administrators can construct and deploy a system "shell", then ask the users to help the system "converge" to the correct parameter values. This way, the enourmous burden of system developments is lifted from the system administrators and spread "thinly" over a multitude of users. The following example illustrates the idea underlying our approach:

Example 1: Consider the development of the data integration system in Figure 1. Currently we must start by building the source schemas and the mediated schema. Next, we must create the semantic mappings among the schemas. This would be the bare minimum that is necessary to allow the functioning of a data integration system (with the help of a query processing engine such as the one described in (Ives *et al.* 1999)). Notice that any of these three tasks is well-known to be difficult and time consuming. For example, even with the help of semi-automatic schema matching tools (Rahm & Bernstein 2001), it is still very labor intensive to manually verify and correct *all* the semantic mappings that the tools suggest, to ensure the correct functioning of the system.

In our approach we can also start by building the source schemas and the mediated schema. Next, we treat the semantic mappings for the mediated schema elements as *system parameters*. We assign "initial values" to the parameters, using random assigments or a semi-automatic schema matching tool (Rahm & Bernstein 2001). Thus, we now have a system "shell": a functioning, albeit likely to be incorrect, system. Next, we deploy the system "shell" on the Internet, and ask users to start using it and providing feedback. We use the user feedback to readjust the values of system parameters, until these values converge.□

In the above example we have focused on treating only semantic mappings as system paramters. However, we believe the approach can also be extended to "learn" other system features, such as the source schemas. Note also from the above example that the mass collaboration approach would not replace, but rather complement well existing techniques to automate specific tasks in building data integration systems (e.g., schema matching and wrapper construction). In fact, we believe it would *amplify* the effects of the current techniques. Finally, the approach would be applicable to building systems in a broad variety of settings, including enterprise intranets, scientific domains (e.g., bioinformatics), and the Internet.

As described, the mass collaboration approach has the potential to dramatically reduce the cost of building data integration systems. But it also raises numerous challenges. In the next section we discuss the challenges and outline the solutions. We then describe the current status of our research in this direction, and preliminary experimental results that show the promise of the approach.

The Mass Collaboration Approach

Our goal is to build a data integration system D. We assume the system administrators have constructed D such that everything necessary has been done, except for a set of parameters $\mathcal{P} = \{P_1, \cdots, P_n\}$ whose values we must set. In Example 1, for instance, everything has been done except for the semantic mappings, which form the set of parameters $\mathcal{P}$.

Our task therefore is to elicit user feedback to help us set the values of $\mathcal{P}$. As users query and interact with the system D, they provide feedback. Periodically, the system will combine all user feedback that has been accummulated so far, to arrive at a new parameter configuration $\mathcal{P}'$. Our assumption is that with sufficient user feedback the system will eventually converge to the correct parameter configuration $\mathcal{P}*$.

We now describe the challenges that arise in applying the mass collaboration approach, and outline the solutions.

System Parameters: The first question we must decide is what we should take to be the system parameters. As mentioned earlier, the parameters can be for example the semantic mappings for the elements of the mediate schema. If we have 10 mediated schema elements then this would yield 10 parameters. The correct value for each parameter is then the correct semantic mapping for the mediated-schema element represented by that parameter. In general, the parameters would be application-specific, and can potentially be anything that we must decide during the process of building the data integration system.

Setting Initial Values of Parameters: The parameters can be set randomly, or initialized using a semi-automatic tool (e.g., any of the many schema matching tools (Rahm & Bernstein 2001)). We believe the closer the initial values are to the correct ones, the sooner the system will converge.

Starting with a Partially Correct System: If we begin by setting *all* system parameters in the manner described above, it is very likely that we would obtain an initial *incorrect* system. But users are unlikely to want to use such a system, because querying it would produce incorrect results. To address this problem, we propose to start with a *correct "subsystem"*, to make sure that users can immediately obtain some value from interacting with the overall system.

To continue with the book example, we would begin by *decoupling* the mediated schema and the query interface. We keep the mediated schema intact, but start with a simple query interface that has only two attributes: title and price. Next, we manually find the correct mappings for these two attributes (over all sources in the system). This would immediately yield a correct, albeit simple, data integration system that allows users to query for book titles and prices. We then leverage user feedback to learn the correct mappings for *other* mediated-schema attributes, such as authors, publishers, and rating.

Once we have learned the correct mapping between a mediated-schema attribute and some sources, we immediately add that mediated-schema attribute to the query interface, to allow the user the possibility of querying also over that attribute. This way, we can gradually expand the query interface – thus the capabilities of the data integration system – but ensure that querying over the query interface always produces correct results.

Enticing Users to Give Feedback: This has usually been considered one of the most difficult problems facing mass collaboration approaches. We propose several ways to elicit feedback:

- *Forced Feedback:* Every time the user asks a query, we make him or her "jump through a hoop". The hoop is a diaglog box with a simple question to which the user answers by clicking an "yes", "no", or "not sure" button. (We discuss the types of questions below.) We think about this as a "capitalist" way of building and maintaining systems. The user *uses* the service of the system, hence he or she should *"pay"* for it, and the "payment" here is *a bit of user knowledge*, in order to help build and maintain the system.

"Hoop jumping" (i.e., forced feedback) should not be used frequently. For example, we can elect to ask the user to jump through the hoop once every five queries, rather than once every single query. We should also try to make sure that the system is so compelling to use that the user is willing to "pay", that is, to put up with the "harassment". This can happen if the system provides value-added services as compared to alternative systems (this is analogous to people's willing to pay more at *amazon.com* for a better customer service). This suggests that if possible we should begin with a system that people already genuinely want to use, then gradually build in mass-collaboration feedback mechanism, in order to expand system capabilities.

- *Volunteer Feedback with Instant Gratification:* Once the user has asked a query, the system produces the results, but also indicates to the user that *even more details* about the results can be provided, if the user is interested. To get to those details, however, the system needs the user to provide some feedback. Thus, the user has a strong incentive to supply some simple feedback, because the feedback provides *instant gratification* in terms of more details about the answers.

  For example, suppose the current data integration system in the book domain has a query interface that allows users to query on book title and price. Suppose further that a user has queried it to find all books whose title contain the phrase "data integration" and whose price is under \$100. The system executes the query, and displays the listings of desired books in terms of a table with several columns. The first two columns are title and price, which the system knows how to fill (because it already has the semantic mappings from title and price in the mediated schema to those in the source schemas).

  The third column is about publishers, but the system does not know how to fill because it does not have any semantic mapping for publisher yet. Thus it populates this column with question marks. Suppose now that the user wants to find out the publisher for a certain book in the result table. Then the user can click on the question mark that represents the publisher for that book. The system will ask the user a few questions, in order to find out which attribute of the source (that contains the book) would map to publisher. Once this has been decided, the publisher fields of all books in that source can be filled with the correct values, and the system has also learned a correct semantic mapping during the process.

- *Volunteer Feedback with Delayed Gratification:* In certain domains, the users may be willing to provide feedback if they know that such feedback will bring long-term benefits. For example, a development team (of say, 10 or 12 people) can work collaboratively to build a data integration system, without having any immediate gratification. Within an organization intranet, the employees may understand the long-term benefits of providing feedback and are willing to do so, to help build systems over the organizational data. Bioinformatists may want to collaboratively build a data integration system over the hundreds of bioinformatics sources on the Internet, and thus may be willing to provide feedback without any immediate benefits.

  Users may also volunteer to contribute feedback to *teach* the system. We can also institute a mechanism of distributing credits (or even monetary payment) to contributors, similar to those employed by many collaborative Web sites, such as *epinions.com* and *amazon.com*.

A principle we follow is that users who "pay" more (either with forced or volunteer feedback) must be able to get higher quality service from the system. Those who pay nothing would get the plain vanilla service.

The Types of Questions That Users Are Being Asked: These questions can be at different granularities. We can display a simple data instance and ask the user to recognize if it is a book title, a name, a publisher, or none of the above (thus treating him or her as a recognizer (Doan, Domingos, & Halevy 2001)). If the user recognizes say five data instances of a source-schema attribute to be publishers, then the system can conclude with high probability that that attribute is about publishers.

We can also display the name of a source-schema attribute, together with several of its data instances, and ask the user to recognize the attribute directly. Whether this type of question would demand more cognitive load from the user than the previous type of question is a subject of our current research.

In general, we believe that forced feedback (i.e., hoop jumping) must be as *cognitively simple* as possible, so that the user has to spend only a minimal amount of time on it. Other types of feedback, because the user "volunteers" to do them, can be more cognitively complex.

Handling Malicious and Ignorant Users: We require that users register and log in to use the system (this is not really a hassle because a user only has to log in for the first time, cookies can take care of the subsequent sessions). This allows us to monitor user activities and compute a weight value that reflects how much we trust the feedback of a particular user. The weight is computed from user feedback on a few "training" sources whose semantic mappings we already know.

To prevent softbots from registering en masse and overwhelm the system, we can use a simple Turing test at the registering time to distinguish human users from softbots (similar to those used by Web services such as *paypal.com*).

Combining User Feedback: Periodically the system will combine user feedback to arrive at a new value configuration for the system parameters. The combination will use the user weights. Notice that, in essence, we can treat each user as a *learner*, that is, classifiers that have been trained and are ready to make predictions on the system data and attributes. This immediately suggests the applicability of schemes to combine learners' predictions, as described in several recent works (Doan, Domingos, & Halevy 2001; Doan *et al.* 2002; Do & Rahm 2002; Madhavan *et al.* 2003).

Quantity of Feedback: What happens if each user uses the system only a few times per year? In this case there will not be enough feedback to adequately learn the weight of each user, and thus to combine user feedback. While this is a valid concern, we believe in practice there are many settings where users frequently use the system. Furthermore, we conjecture that user usage often follow a Zipfian distribution, with a small number of very active users and a large number of infrequent users. For example, many of us use services such as *amazon.com* and *epinions.com* infrequently, but they are still full of user feedback and contribution, which suggests a substantial number of active users. Our simulated experiments (discussed in the next section) suggest that the system can zoom in on users with high quality feedback, and that it can converge in a reasonable amount of time even with a small number of such users.

Impact of Feedback: Another concern is whether the impact of *each* feedback would be too small to even make a difference. We note that this is not the case in terms of learning many types of system parameters. Experiments such as those in (Perkowitz *et al.* 1997) and our work (Doan, Domingos, & Halevy 2001) suggest that only a small number of correct feedback on the data instances of an attribute is necessary to learn the correct semantic mapping of that attribute with high probability. We have also mentioned that feedback can be solicited at different granularity level. Thus, a single feedback can also decide the semantic mapping of an attribute.

Enticing Users to Use the System: How can we make sure that the user wants to use our system, and not an equivalent system that is manually constructed but does not "harass" users with any feedback mechanism? Our solution is to make sure that our system would "subsume" the manually constructed system or at least a significant portion of it. This way, users who do not want to give feedback could still use our system but only at the service level of the manual version.

On the other hand, users who is willing to give feedback will get access to the more advanced version of the system. Since the system can leverage the feedback to continuously improve its services, those who give feedback will get access to ever improving services.

Current Status of Our Work

Besides developing a general framework for applying mass collaboration to build data integration systems, we are currently testing our ideas using simulation as well as real-system deployment.

We have simulated the interaction of a broad variety of user populations with data integration systems. As an example, a specific scenario of our simulation has a population of 5000 users, with user quality randomly selected over the interval [0,1]. A user of quality p gives the correct answer with probability p. The data integration system has a mediated schema of 10 attributes, and consists of 10 sources, each of which also has 10 attributes. In this scenario, the system needs an average of 14 feedback answers *per user* to converge to the correct semantic mappings for all mediated-schema attributes (over all sources).

We are also currently building a real-world comparison shopping system with a feedback mechanism. We plan to use the system to evaluate the participation of real users. We shall start with volunteers to evaluate if users can in fact reliably handle the cognitive load of answering system questions. For more details on the current status of our work, see (McCann *et al.* 2003).

Related Work

Our work draws from many related areas, which we discuss below.

Knowledge Base Construction via Mass Collaboration: Our work was inspired by several recent works that attempt to leverage the large volume of Web users to build knowledge bases and tech support websites ((Richardson & Domingos 2003; Richardson, Aggrawal, & Domingos 2003), *quiq.com, openmind.org*). The basic idea of these works is to have users contribute facts and rules in some specified language. Our work differs from these in several important aspects. First, in building a knowledge base, potentially *any* fact or rule being contributed constitutes a parameter whose validity must be checked. Thus, the number of parameters can be very high (potentially in the millions) and checking them poses a serious problem. In contrast, the number of (system) parameters in our case is comparatively much smaller and thus potentially much more manageable. Second, such knowledge bases must provide some mechanisms to allow users to immediately leverage the contributed information (to gain some instant gratification effect). Providing such mechanisms in the context of knowledge bases can be quite difficult, because it requires performing inference over a large number of possibly inconsistent or varying-quality facts. Such mechanisms are considerably much simpler in our case, because feedback on the system parameters can immediately affect the query results.

Building Data Integration Systems: The manual construction and maintenance of data integration systems is very labor intensive and error prone. There have been many works on reducing the labor costs of *specific* tasks during the construction process, such as schema matching (Rahm & Bernstein 2001) and wrapper construction (e.g., (Kushmerick, Weld, & Doorenbos 1997; Ashish & Knoblock 1997)), but few works on a systematic effort to address cost reduction for the *whole process*, with the exception of (Rosenthal *et al.* 2001; Rosenthal & Seligman 2001). Our work on mass collaboration can be seen as providing a systematic solution to this problem.

Semantic Web: Our work shares several common issues with research on the Semantic Web, such as enticing users to provide feedback and combining information of varying quality. The idea of instant gratification is articulated in (McDowell *et al.* 2003; Etzioni *et al.* 2003).

Machine Learning: We have mentioned that each contributor in the mass collaboration framework can be thought of as a learner (which has been trained and is ready to make predictions). We believe the issue of how to learn the accuracy of learners and combine a very large number of learners in an efficient and accurate way raises interesting learning issues that have not been considered before, and thus may warrant further studies.

Schema Matching: Numerous works have been conducted on schema matching, a fundamental problem in integrating data from heterogeneous sources. Some recent works include (Milo & Zohar 1998; Palopoli, Sacca, & Ursino 1998; Li & Clifton 2000; Madhavan, Bernstein, & Rahm 2001; Doan, Domingos, & Halevy 2001; Yan *et al.* 2001; Kang & Naughton 2003; He & Chang 2003; Madhavan *et al.* 2003) (see (Rahm & Bernstein 2001) for a survey). These works employ manually crafted rules and machine learning techniques, with some limited human interaction, to discover semantic mappings. In contrast, our current work leverages the feedback of a multitude of users to find the mappings. To our knowledge, this is the first work on schema matching in this direction.

In the current work, we have focused only on finding one-to-one mappings, such as "location maps to address". We note that even this problem setting is already very difficult. The vast majority of schema-matching works have focused only on this problem (Rahm & Bernstein 2001). We are currently extending our framework to find more complex mappings, such as "location maps to the concatenation of city and state" and "price maps to listed-price * (1 + tax-rate)".

Autonomic Systems: Our work here is also related to autonomic systems in that data integration systems in the mass collaboration scheme can also exhibit autonomic properties such self-healing and self-improving. The key difference is that autonomic systems have traditionally been thought of as achieving these properties by observing the external environment and adjusting themselves appropriately. In contrast, our systems are observed by the external environments (i.e., the multitude of users) and then are adjusted by them accordingly.

Conclusion

The current cost of ownership of data integration systems is extremely high, due to the need to manually build (and maintain) such systems. In this paper we have proposed a mass collaboration approach to efficiently build data integration systems. The basic idea is to shift this enourmous cost from the producers (of the system) to the consumers, but spread it "thinly" over a large number of consumers. We have discussed key challenges of this approach and outlined the solutions. We have also described the current status of our research in this direction, and discuss the relationship between this work and several other areas. This research is conducted within the context of the AIDA (Automatically Integrating DAta) project at the University of Illinois, whose goal is to build autonomic data integration systems.

References

Arens, Y.; Hsu, C.; and Knoblock, C. 1996. Query processing in the SIMS information mediator. In Tate, A., ed., *Advanced Planning Technology*. AAAI Press.

Ashish, N., and Knoblock, C. 1997. Wrapper generation for semi-structured internet sources. *SIGMOD Record* 26(4):8–15.

Avnur, R., and Hellerstein, J. 2000. Continuous query optimization. In *SIGMOD '00*.

Chen, J.; DeWitt, D.; Tian, F.; and Wang, Y. 2000. Niagaracq: A scalable continuous query system for internet databases. In *SIGMOD '00*.

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.; Madhavan, J.; Domingos, P.; and Halevy, A. 2002. Learning to map ontologies on the Semantic Web. In *Proceedings of the World-Wide Web Conference (WWW-02)*.

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*.

Duschka, O. M., and Genesereth, M. R. 1997. Query planning in infomaster. In *12th ACM Symposium on Applied Computing*.

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*.

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).

Haas, L. M.; Kossmann, D.; Wimmers, E. L.; and Yang, J. 1997. Optimizing queries across diverse data sources. In *Proc. of VLDB '97*.

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

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)*.

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.

Kwok, C., and Weld, D. 1996. Planning to gather information.

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

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.

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.; Bernstein, P.; Chen, K.; Halevy, A.; and Shenoy, P. 2003. Matching schemas by learning from a schema corpus. In *Proc. of the IJCAI-03 Workshop on Information Integration on the Web*.

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)*.

McDowell, L.; Etzioni, O.; Gribble, S.; Halevy, A.; Levy, H.; Pentney, W.; Verma, D.; and Vlasseva, S. 2003. Evolving the semantic web with mangrove. Technical Report UW-TR-2003, Dept. of CSE, Univ. of Washington.

Milo, T., and Zohar, S. 1998. Using schema matching to simplify heterogeneous data translation. In *Proc. of VLDB*.

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.

Perkowitz, M.; Doorenbos, R.; Etzioni, O.; and Weld, D. 1997. Learning to understand information on the Internet: An example-based approach. *J. Intelligent Information Systems* 8(2):133–153.

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

Richardson, M., and Domingos, P. 2003. Building large knowledge bases by mass collaboration. Technical Report UW-TR-03-02-04, Dept. of CSE, Univ. of Washington.

Richardson, M.; Aggrawal, R.; and Domingos, P. 2003. Building the Semantic Web by mass collaboration. Technical Report UW-TR-03-02-05, Dept. of CSE, Univ. of Washington.

Rosenthal, A., and Seligman, L. 2001. Scalability issues in data integration. In *Proceedings of the AFCEA Federal Database Conference*.

Rosenthal, A.; Renner, S.; Seligman, L.; and Manola, F. 2001. Data integration needs an industrial revolution. In *Proceedings of the Workshop on Foundations of Data Integration*.

Yan, L.; Miller, R.; Haas, L.; and Fagin, R. 2001. Data Driven Understanding and Refinement of Schema Mappings. In *Proceedings of the ACM SIGMOD*.

Yerneni, R.; Papakonstantinou, Y.; and Garcia-Molina, H. 1998. Fusion queries over internet databases. In *Proc. of the 6th Int. Conf. on Extending Database Technology (EDBT)*.