Collective Data Integration for Virtual Organizations

Robert McCann, Warren Shen, AnHai Doan
Department of Computer Science
University of Illinois at Urbana-Champaign, IL, USA
{rlmmcann,whshen,anhai}@cs.uiuc.edu

Abstract

When multiple parties join forces in a "virtual organization" to reach a common goal, they often need to integrate disparate data sources. Such integration differs from traditional integration settings in at least two important aspects. First, most virtual organizations do not have highly trained experts to carry out the integration process, as "physical" home organizations often do. Hiring such experts is difficult given the short-lived nature of virtual organizations. Second, in virtual organization settings the need for short-lived, on-the-fly integration often arises. Current integration techniques are geared toward building long-running integration systems, and hence are not well suited to fulfill such needs.

This paper outlines a solution to the above problems. The solution enables all participants in the virtual organization to collectively contribute to the process of integrating the data sources. The contribution can range from answering a few very simple questions to solving more cognitively challenging tasks. This way, motivated, but "data integration-illiterate" participants can also make contributions, thus significantly reducing the need for experts. To address the need for on-the-fly data integration, we discuss how the above solution can be adapted so that each user with an on-the-fly data integration query needs only to answer a few simple questions, to obtain the answer to the query.

1 Introduction

A *virtual organization* is a dynamically formed coalition of parties from different home organizations, who want to cooperate with one another to reach a common goal. Prime examples of such organizations arise in business enterprises, government agencies, scientific projects, emergency situations, and homeland security. Very often, participants in a virtual organization must integrate information from disparate data sources of the home organizations. Such integration differs from traditional integration settings in at least two important aspects.

Serious Lack of Integration Experts: Data integration is well known to be a complex and labor intensive process [17, 6, 11, 7, 9, 4, 3, 10]. Hence, home organizations often must recruit *highly trained experts* to build *data integration systems*, then retain the experts to maintain them over time. Typically, a "physical" home organization can hire only a small team of such experts, due to the difficulties in training and the high cost of hiring.

Since each home organization can participate in many virtual organizations (*e.g.*, see Section 2), it is very difficult for the small team of experts to participate in all such virtual organizations, in addition to serving the integration needs of the home organization. To make matters worse, many home organizations (*e.g.*, hospitals, fire departments) may contribute valuable data, but have no data integration experts. As a consequence, virtual organizations often have no or very few data integration experts at their disposal. The serious lack of such experts makes it extremely difficult or practically impossible for virtual organizations to build and maintain data integration systems.

Strong Needs for On-the-Fly Integration: To date, data integration (at home organizations) has typically been limited to *long-running* information needs, to amortize the high cost of constructing and maintaining data integration systems. Examples of such information needs arise in integrating data of merging companies, querying Deep Web data sources, building digital governments, and integrating bioinformatics data sources.

While many virtual organizations do need to build and maintain data integration systems, due to their

dynamic nature, such organizations also have strong needs for short-lived, on-the-fly data integration. For example, a participant p from home organization A may need to ask only one specific query over three data sources S_1, S_2, and S_3 of home organizations B and C.

Current integration techniques are not well suited to handle this on-the-fly integration need for three reasons. First, they are designed to build a data integration *system* over sources $S_1 - S_3$, not for answering a specific *query*. Second, building such a system often takes days or weeks, while p may need to obtain the answer to the query today. Finally, even if p is strongly motivated to use current integration techniques to obtain the answer, he or she is unlikely to be able to do so, given that such techniques are still designed only for integration experts. Thus, it has now become crucial to develop tools that enable even "integration-illiterate" participants to perform quick, on-the-fly integration of data.

Collective Data Integration: In this paper we propose a solution to address the above problems. At the heart of our solution is the idea that all participants of a virtual organization can collectively contribute to the process of building a data integration system, thereby (a) significantly cutting the cost, (b) rapidly deploying the system in time for the organization needs, and (c) reducing the need for hiring data integration experts.

Briefly, we propose to decompose each data integration task (*e.g.*, wrapper construction, schema matching, etc.) into "bite-size" chunks, such that each participant can solve the chunks based on his or her time and effort constraints and abilities. The contribution can range from answering a few very simple questions to more cognitively challenging tasks. This way, motivated, but "data integration-illiterate" participants can also make contributions.

The solution thus has an open-source flavor. For example, in the *Linux* community participants collectively build the *Linux* operating system. Here, the participants of a virtual organization collectively build a data integration system. We call this solution MOBS, which is shorthand for "Mass Collaboration to Build Systems".

We then discuss how MOBS can be adapted such that a participant can solve his or her on-the-fly integration needs by answering a series of questions. Finally, in the same spirit of collective contribution, we discuss how a current on-the-fly integration need can be fulfilled by leveraging answers from multiple users to previous on-the-fly integration needs.

In several other papers [12, 13], we have described MOBS in detail, as a general layer on top of current integration techniques, to significantly reduce the cost of building data integration systems. In this paper we focus on motivating and describing why MOBS is well-suited to the virtual organization context.

2 A Motivating Example

Our recent interaction with the Illinois Fire Service Institute (IFSI, *www.fsi.uiuc.edu*) highlights data integration challenges for virtual organizations and motivates the needs for mass collaboration techniques. Among its many responsibilities, IFSI assists Illinois fire departments in developing fire fighting programs. For example, suppose a fire department A must develop an emergency program to handle bio-hazardous situations. It contacts IFSI with the request.

IFSI will broadcast the request, then assemble a loose coalition of organizations – that is, a *virtual organization* – to help A. The coalition may contain hospitals, police departments, fellow fire departments, fire service institutes in other states, and so on. Once the coalition is created, A can ask questions and obtain answers, all mediated via IFSI.

First, A probes hospitals, police departments, and other fire service institutes to understand various bio-hazardous situations (*e.g.*, the frequencies, severities, consequences, etc.). Next, A asks fellow fire departments if any of them has developed similar emergency programs, and if so, A may request information about those programs. The coalition is dissolved when A indicates it no longer needs outside help in developing its own emergency program.

IFSI plays the intermediary role because it is in the best position to assemble a coalition to help A. Organizations in the coalition want to help A. They also benefit from such help because they can gain a deeper understanding of bio-hazardous situations and emergency program solutions at other places.

To date, IFSI executes such coalition building and data exchange mostly by hand. Unfortunately, it is

rapidly running out of human resources to keep up with the demand in the post-9/11 world. Hence, it has turned to our data integration group at UIUC for help.

Currently, our group is working with IFSI to train in-house data integration experts, as well as building tool kits to help these experts rapidly construct data integration systems. For example, suppose that a coalition consists of four fire departments, each with a database of fire fighting materials. The experts at IFSI can quickly construct a virtual data integration system over the four databases. The system provides a uniform query interface (also known as a *mediated schema*) over the database sources, thus enabling A to rapidly query the sources and obtain answers, and saving IFSI the tedious task of executing A's query at each individual source.

Training data integration experts for IFSI, however, can provide only a partial and short-term solution. In the long term, IFSI experts will not be able to keep up with the numerous coalitions that must be formed to satisfy information requests of the client departments. IFSI is actively seeking funding, but even in the best-case scenario it probably cannot maintain more than a few experts. To address this problem, we propose a mass collaboration solution, which we describe next.

3 The MOBS Solution

We now briefly describe the basic ideas behind MOBS (see [12, 13] for more detail). In the next section (Section 4) we discuss why MOBS is particularly well-suited for virtual organizations. Section 5 comments on past successes using MOBS to deploy data integration systems and Sections 6 and 7 discuss adapting MOBS specifically for on-the-fly data integration.

Consider again the virtual data integration system over the four fire departments described in the previous section, which allows users to query for fire fighting materials. For ease of exposition, assume that the system's mediated schema has only five attributes $year, equipment, category, manufacturer$, and $price$, and that the component database sources have schemas S_1, S_2, S_3, and S_4, as shown in Figure 1. Building such a system involves many difficult tasks [1], one of which is *schema matching*:

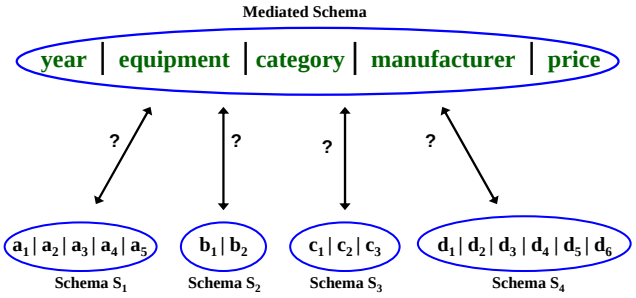

Figure 1: Using mass collaboration to find the semantic matches between the attributes of the mediated schema and the source schemas.

finding the semantic correspondence between the attributes of the mediated schema and those of the source schemas [15, 5].

In what follows we illustrate MOBS by describing how it can assist the integration experts in accomplishing this task.

To apply MOBS, the experts begin by building a small data integration system: they manually specify the correct matches from *year* to each of the source schemas. Suppose these matches are $year = a_1, year = b_1, year = c_1$, and $year = d_1$. These matches yield a system whose query interface consists of a *single* attribute: *year*. Users can immediately query this system to find materials based upon year. Note that building the small data integration system is still much less work than building the complete system. Further, in Section 4 we show that in many cases the experts can leverage other so-called "helper" applications and do not need to build initial small data integration systems.

Decomposing the Problem: Next, the experts must find the correct matches for the remaining four mediated-schema attributes. They decompose this problem into a series of *questions*, by considering *all* pairings between mediated-schema attributes with source-schema ones: $equipment = a_1, equipment = a_2, \ldots, equipment = a_5, equipment = b_1, \ldots, price = d_6$. A question such as $equipment = a_1$ means "Does *equipment* match a_1?". Knowing the correct answers to the above questions amounts to solving the schema matching problem.

Soliciting User Answers: The experts now deploy the above small data integration system and ask users to start using it. When a user submits a

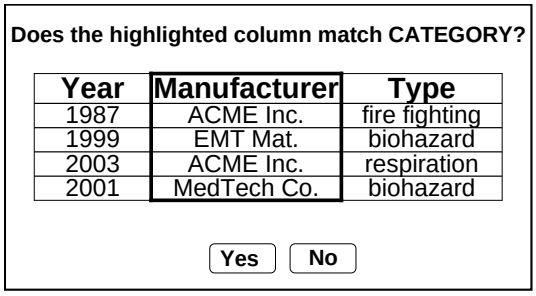

Figure 2: A sample question that the system asks the user to answer.

query to the system (*e.g.*, *"Find all materials bought within the last two years."*), the system poses a simple question to the user. Only after the user has answered the question would the system display the query result. Thus, in essence, the system makes the user "pay" for the question.

Figure 2 shows a sample question that asks the user whether attribute named $manufacturer$ of a source (say, attribute d_3 of source S_4) matches attribute *category* of the mediated schema. The user will answer "yes" or "no", after examining the *name* of the attribute, several of its *data instances*, and the *context information* (*i.e.*, the surrounding attributes in this case).

Assessing User Reliability: To handle malicious or ignorant users, the system computes a *weight* for each user that measures the quality of his or her answers. Earlier the experts have set aside some sources, say S_1, for *user evaluation*, and have manually provided all correct matches for these sources.

Now the system can ask users *evaluation* questions that are related to this source: $year = a_1, year = a_2, \ldots, year = a_5, equipment = a_1, \ldots, price = a_5$. Since the system already knows the answers to these questions (because it knows the manual correct matches), it can evaluate user answers and compute their weights.

Consider a user u_1. If the number of answers that u_1 has provided on the parameters coming from the evaluation sources is below a threshold k, then we say that user u_1 has not provided sufficient answers in order to be evaluated, and set $weight(u_1) = 0$. Otherwise we set $weight(u_1)$ to be the fraction of u_1's answers that are correct.

To track users we require them to login to use the system. Note that a user needs to login only once; subsequent sessions can be handled automatically by cookies. For any single user we randomly mix the evaluation questions with teaching questions (*i.e.*, questions used to get feedback on unlabeled sources) to ensure that the user does not know which ones are the evaluation questions. We call a user *trustworthy* if his/her weight is above a threshold w (currently set at 0.65) and *untrustworthy* otherwise.

Combining User Answers: Now consider a question such as $category = b_1$, which will receive a steady stream of "yes" and "no" answers from trustworthy users. The system monitors this stream of answers; as soon as a *convergence criterion* is satisfied, it assigns to $category = b_1$ a combined answer based on the answers it has received so far, and stops soliciting further answers for this question.

Suppose $category = b_1$ has received a total of n answers. The current convergence criterion is to stop if (a) the number of majority answers (either "yes" or "no") reaches $n * 0.65$ and n exceeds 20, or (b) n exceeds 50. In either case the system returns the majority answer as the final answer for $category = b_1$. It then proceeds similarly with other questions. Once the system has obtained answers to all questions, it has found all correct matches between the mediated schema and source schemas.

In [13] we describe a solution that leverages the above idea and employs dynamic Bayesian Network techniques [16] to combine user answers in a principled way.

4 Applying MOBS to Virtual Organizations

We now consider applying the above MOBS idea to virtual organizations.

4.1 Task Decomposition

The first problem to consider is how to decompose a data integration task (*e.g.*, schema matching, wrapper construction, query translation, etc.) into smaller chunks that can be distributed to a set of people. If these people are ordinary, "integration-illiterate" users, then the chunks have to be very simple, such as the yes/no questions that we described in the previous section. If the people are integration experts or trained programmers, then the chunks could

be more coarse grained, and cognitively demanding (*e.g.*, writing source code).

It is likely that not all data integration tasks can be decomposed into a set of simple questions, but we show in [13] that many tasks can be decomposed. Further, even if only a part of a task can be decomposed, we believe that can already result in a significant reduction in labor cost.

4.2 Incentive Models

We now discuss how to entice users to provide feedback. First, we discuss two models of prior work, and show how they can be adapted to our context. Then we propose two novel models that address some important limitations of prior models.

Volunteer/Delayed-Gratification Model: The vast majority of works and real-world applications in mass collaboration rely on users volunteering feedback. Examples include Bibserv [2], *openmind.org*, *amazon.com*, and *epinions.com*. Such volunteering is typically motivated by some (often delayed) benefits or self-gratification. This model clearly also applies to our virtual organization context. For example, the participants of a virtual organization understand the benefits of providing feedback and are willing to help build systems over organizational data.

Instant Gratification Model: Some recent works propose to provide users with *instant* benefits as soon as they volunteer some feedback, on the ground that such immediate gratification may motivate users to volunteer more feedback. The instant benefits could be value-added services [14] or game play experience, which helps implicitly generate user feedback (*e.g.*, *espgame.org*).

This model can be easily adapted to our context. For example, a system in the fire fighting domain would produce the usual list of results in response to a user query, but would also indicate that *even more details* about those results are available, if the user is interested. To get to those details, however, the system needs the user to answer a simple question. Thus, the user has a strong incentive to supply some simple feedback, which provides instant gratification in terms of more details about the query results.

The above two models rely on users being *willing to provide feedback*. While we believe this assumption is generally valid in virtual organization contexts, we propose to also consider two models in which users are *forced to participate*, hence guaranteeing a steady stream of user answers for the mass collaboration process.

Monopoly/Better-Service Model: If the application under consideration provides a monopoly service that its users need, or a service in much better ways than its competitors, then it can leverage that advantage to "force" the users to "pay" for using its service, where the "payment" is a bit of user knowledge. For example, consider an internal web service that the employees of an organization must access daily. Since this internal web service is in a monopoly situation, it can use MOBS to collect "payments" from the employees, then leverage these "payments" to improve its services.

Helper Application Model: A data integration system can also leverage the users of *another application* – which we call a *helper application* – to provide feedback. For example, within an organization, a data integration system can leverage the users of an internal web service (as described above). For this model to work, we believe the helper application must be providing a monopoly or better service. Helper applications are in a sense similar to the initial capitals invested in the real world in a business, until the business grows to the extent that it can sustain itself.

5 Empirical Evaluation

We have applied MOBS to build data integration systems in several settings. For example, we enlisted the students of several data management courses to build data integration systems in the online bookstore domain and the Computer Science Department domain. The number of students in those applications ranged from 26 to 132, respectively. We also enlisted our research group (approximately 12 persons) to build a bibliography of data integration papers on the World-Wide Web.

In all experiments, each participant needed to spend very little effort (*i.e.*, answering on average fewer than three yes/no questions per day) to rapidly build the systems over several days. In addition, the labor costs of the experts were reduced significantly, in that they spent very little time setting up MOBS

for these systems.

We have also carried out extensive simulation experiments, which support the above results, and further examine the MOBS solution. A detailed description of the empirical evaluation is in [13].

6 On-the-Fly Integration

The previous discussions outlined the process of building a traditional data integration system for use by virtual organizations. However, given their short-lived nature, virtual organizations often require on-the-fly integration services. On-the-fly data integration is an emerging topic ([8]) in which the system need only provide one-time answers to a fixed set of queries. In this setting the integration system is not necessarily persistent and the query capabilities must only support a fixed set of queries.

Consider again our fire fighting example from Section 2. Members of fire department A seek information in order to build a biohazard response program. Thus a particular member p of A may only be interested in a small set of queries such as *"Which biohazard equipment manufacturer is most popular in other departments?"* and *"What percentage of equipment budgets are spent specifically for biohazards?"*. Furthermore, only a one-time answer to these queries is required.

In such a setting we only need to perform the data integration steps necessary to answer these particular queries. For the example illustrated in Figure 1 we only need to perform schema matching for the mediated schema attributes $manufacturer$, $category$, and $price$ over fire department sources.

The MOBS solution outlined in Section 3 is directly applicable to this setting. To perform on-the-fly integration, we can apply MOBS to build a "mini system" on demand for a given set of queries. A "mini system" is a temporary, small-scale data integration system constructed over a subset of all sources and mediated schema attributes. In our current example, person p is asked questions only relevant to building their "mini system" over the three mediated schema attributes $manufacturer$, $category$, and $price$. This system is kept confidential and only provided to this particular person.

There are three significant advantages to this approach over traditional data integration (Section 3). First, the size of this "mini system" is smaller, hence less time and work are needed to construct the system. This speed is crucial for meeting the requests of each virtual organization member in a timely fashion.

The second advantage is that each "mini system" is private. If a person provides incorrect answers and builds a faulty "mini system", they hinder only their own cause. Since there are no global effects in this case (*e.g.*, constructing a faulty global integration system), MOBS can be deployed to converge quite rapidly and employ little or no user evaluation.

Finally, the content of this "mini system" is relevant to the associated virtual organization member. Given that each system is built to satisfy a particular set of queries, we can presume that the person asking these queries is somewhat knowledgeable about the attributes and sources being integrated. In our example, person p can be expected to know what a fire department is, what categories of equipment are used by such departments, and even who are the principle equipment manufacturers. Given this familiarity, we can expect that p is capable of accurately answering questions and hence successfully building their "mini integration system" on the fly.

7 Reusing On-the-Fly Integration

A challenge highlighted by the previous section is to *quickly* perform on-the-fly integration. It is true that "mini systems" constructed on the fly may be small (compared to more general global systems) and that MOBS can be tuned to converge these systems quite rapidly. However, these "mini systems" are constructed from scratch as needed. A potentially large improvement is to leverage the work of past individuals to more quickly meet the needs of each new individual.

For this challenge we again employ the MOBS framework (Section 3). First we archive all "mini systems" constructed over all virtual organizations. Given a new "mini system" M, we can leverage the archive in order to more quickly build M as follows. Suppose we must learn the value of statement $category = b_1$ and we find this statement several times within the archive. In each of these archived instances this statement is set to some answer (*i.e.*, *"yes"* or *"no"*). These answers can be immediately

fed into our MOBS framework as votes on the correct value of $category = b_1$. Most importantly, these answers are collected automatically without explicitly asking questions of the new virtual organization member, hence speeding the construction of M.

One issue in the above scheme is measuring the credibility of each archived answer. In the strict on-the-fly setting (the previous section), global trust was not an issue since each "mini system" was constructed in isolation. This is not the case here. Thus we must tune MOBS to reincorporate some combination of user evaluation and statistical convergence. However, we feel that this could be done with small enough overhead that reuse of past "mini systems" still provides a substantial improvement in the time required to satisfy future on-the-fly integration needs.

8 Conclusion and Future Work

By their nature, virtual organizations are usually dynamic and short-lived, and are unlikely to have many integration experts to help them quickly integrate data. Even when experts are available, they still face the problem of having to integrate data in a very short amount of time, for which conventional integration techniques are not well suited.

In this paper we have proposed a mass collaboration solution to this problem. The basic idea is to decompose data integration tasks into small chunks, where the chunks can be as simple as yes/no questions, then leverage the participants of the virtual organization to help solve the chunks, thereby contributing toward building and maintaining data integration systems. Participants in a virtual organization work together toward a common goal, and therefore are likely to be strongly motivated to participate in the mass collaboration process.

As described, our solution has the potential to help virtual organizations quickly integrate data, with far fewer experts and much less workload on these experts. We also show how the mass collaboration solution has the potential to provide on-the-fly integration for needy users, and subsequently leverage these "mini systems" to benefit future integration tasks.

We are further developing the mass collaboration framework, and applying it to real-world contexts, including those of virtual organizations.

References

[1] www.cs.washington.edu/homes/alon/files/btw2003.ppt.

[2] R. Agrawal, P. Domingos, and M. Richardson. Trust management for the semantic web. In *Proc. of the Int. Semantic Web Conf.*, 2003.

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

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

[5] A. Doan, P. Domingos, and A. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In *Proceedings of the ACM SIGMOD Conference*, 2001.

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

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

[8] A. Halevy and C. Li. http://www2.cs.washington.edu/nsf2003/final-reports/idm2003-report.pdf. 2004.

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

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

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

[12] R. McCann, A. Doan, A. Kramnik, and V. Varadarajan. Building data integration

systems via mass collaboration. In *Proc. of the SIGMOD-03 Workshop on the Web and Databases (WebDB-03)*, 2003.

[13] R. McCann, W. Shen, A. Kramnik, V. Varadarajan, O. Sobulo, and A. Doan. Integrating data of disparate sources: A mass collaboration approach. Technical Report UIUC-TR-2004, Dept. of CS, Univ. of Illinois. To appear., 2004.

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

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

[16] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

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