# CAREER: Evolving and Self-Managing Data Integration Systems

AnHai Doan, University of Illinois at Urbana-Champaign

**Project Description**

## 1 Introduction

Data integration has been a long standing challenge for the database community. Indeed, all six "white papers" on future research directions that our community has published (in 1989-2003) acknowledged the growing need for integrating data from multiple sources [15, 136, 137, 135, 16, 3]. This need has now become critical in numerous contexts, including integrating data on the Web and at enterprises, building e-commerce market places, and analyzing data for scientific research [3].

Consequently, much research has been conducted on data integration, and many integration architectures have been proposed [56, 134, 122, 138, 139, 5, 22, 141, 61, 71] (see [55] for detailed surveys). A well-known and important architecture is that of *virtual data integration systems*, which provide a uniform query interface over a multitude of data sources [65, 98, 141, 82, 92, 63, 87, 66, 86]. Over the past decade, such systems have been researched intensively. The bulk of work has focused on architectural and query processing aspects [76, 65, 98, 141, 71, 82, 92, 9, 53, 54, 147, 125, 6, 24], and has laid down a solid foundation on which to build data integration systems. Indeed, if such systems can be now built and deployed widely, they would revolutionize the way we access data [3, 143], and provide a basis on which to build even more advanced information processing systems, such as recently proposed peer data management systems [33, 14, 69] and systems that integrate Web services [113].

Unfortunately, **today data integration systems are still extremely hard to build and costly to maintain**. They must be taught in tedious detail how to interact with the data sources and must constantly be adjusted for changes at the sources (as Section 1.1 will explain). The laborious teaching and adjustment incur huge expenses, and pose a key bottleneck for the widespread deployment of data integration systems in practice. Today, at enterprises, where data integration is frequently a must [3], it is carried out at a tremendous cost, often at 35% of the IT budget [88]. On the Web, where data integration systems can vastly simplify the search for information, there are currently few such systems and at limited scales. In many domains such as fire fighting in rural Illinois, where data integration has been identified by the state of Illinois as crucial for effective community defense [29], it has not been carried out due to the complexity and the high cost involved [130]. The recent advent of languages and mediums for creating and exchanging semi-structured data, such as XML, OWL, and the Semantic Web [145, 13], will further fuel data integration applications and exacerbate the above problem. **Thus it has now become critical to develop techniques that enable the efficient construction and maintenance of data integration systems**.

In this proposal I describe an integrated research and education plan that addresses the above problem. I want to achieve the widespread use of data integration systems, by making them much easier to use, with far less need for human supervision. Hence, **my research goal** is to develop techniques to build data integration systems that *learn to evolve and self manage over time.* As such, this research fits into the emerging paradigm of building computing systems (e.g., databases, storage devices, and Internet services) that manage themselves, motivated by the fact that the complexity of such systems is growing rapidly, and soon can make it impossible for humans to effectively optimize and maintain them in real time. Prime examples of initiatives in this paradigm include autonomic computing at IBM [81], self-tuning databases at Microsoft, IBM Almaden, and others [79, 23, 101, 100, 142], and recovery oriented computing at Berkeley [80]. The research also fits into the grand challenge on conquering the complexity of large-scale information systems that the Computing Research Association (CRA) recently proposed [30].

**My education goal** is to build on the research to prepare students at all levels for the novel challenges posed by data processing in our Internet world, and to educate and engage the public in meeting these challenges. The proposed research and education plan thus lays the foundation for a lifetime career in Computer Science, with a focus on the effective management of distributed and heterogeneous data.

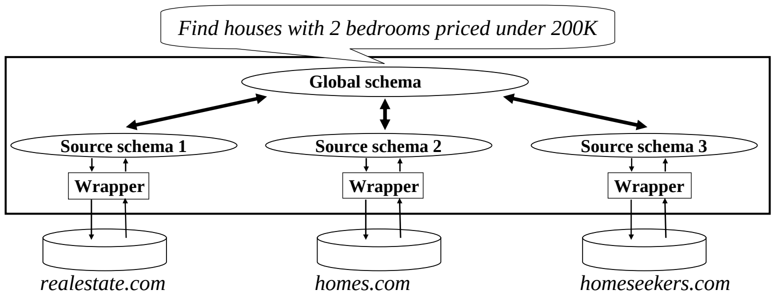

Figure 1: A data integration system in the real-estate domain.

## 1.1 The Problem

I will now describe the architecture of data integration systems, then explain the construction and maintenance problem in detail. Consider a data integration system over three Web sources that list houses for sale, as shown in Figure 1. To construct such a system, the system builder begins by creating a *global schema* that captures the relevant aspects of the real-estate domain. The global schema may contain attributes such as address, price, and description, listing the house address, price, and description, respectively.

Next, for each data source, the builder creates a *source schema* that describes the content of the source, and a *wrapper*, which is a program that knows how to query and extract data from the source via the source's *query interface*. The wrapper also knows how to translate between the source data (e.g., in HTML format) and the data that conform to the source schema (e.g., a set of tuples).

Finally, the system builder create a set of *semantic mappings* that relate the attributes of the global schema to those of the source schemas. (These mappings are shown as bold arrows in Figure 1.) Examples of such mappings are "attribute address of the global schema maps to attribute location of the schema of *realestate.com*" and "price maps to listed-price".

Now given a user query such as "find houses with 2 bedrooms priced under $200K", which is formulated over the global schema, the system can use the semantic mappings to reformulate the query into queries over the source schemas. Next, it optimizes these queries, executes them with the help of the wrappers, then combines the data returned from the sources. Since in practice data sources often contain duplicate items (e.g., the same house listing) [78, 140, 47], the builder frequently must write a program to *detect and eliminate duplicates* from the combined data, before presenting the final answers to the user query.

**The Problem:** As described, when constructing a data integration system, the system builder will have to carry out a set of fundamental tasks, such as global schema creation, wrapper construction, schema matching, and so on. These tasks are well known to be very difficult [10, 126, 3, 90, 37], primarily because they require reasoning about data semantics (e.g., is house-description the same as house-style?). Even though some semi-automatic techniques have been proposed, no satisfactory solution has been found. Hence today the system builder still execute these tasks *largely by hand*, in an extremely labor intensive and error prone process [126, 37].

To make matters worse, in dynamic environments, such as the Web, sources often change their query interface, data formats, or presentation styles [89, 96]. Such changes can invalidate a wrapper or a semantic mapping, causing system failure. Hence, the builder must continuously monitor the deployed system, to detect and repair failures [27]. The prohibitive cost of manual monitoring exacerbates the problem, and makes data integration systems virtually impractical on a Web scale.

## 1.2 The Proposed Solution

**Vision:** To address the above problem, I propose to build data integration systems that learn to evolve and self manage. Imagine that we want to construct a system over 100 real-estate Web sources. Instead of spending months building the system, and having parts of it already stop working (due to changes at the sources) even before the system is completely built, we can start by building only a small system over say, 5, sources. This way we will soon have a system up and running.

The system then *evolves* by expanding to cover new data sources, and eventually will cover all 100 sources. (The system may also *evolve* by "probing" the current sources under its "control" to gather more meta data, such as source latency and quality statistics [114, 116, 60]; but we leave this scenario as a future extension of the current research.) While evolving, the system *maintains* the sources under its "control", by continuously monitoring them to detect and repair failures. The system will still require interaction with humans (i.e., the system builder), but at a far less amount compared to the current practice. Such interaction may be frequent at the beginning, as the system learns more, the amount of interaction decreases.

**Key Idea:** To realize the above vision, a key idea underlying my solution is that the system can learn from *many types of information and entities* in the environment to evolve and self manage. It can learn from past construction activities, data in the domain, other systems, domain knowledge supplied by the builder, behaviors of systems and sources, and even from the multitude of users. For example, when evolving, the system can learn from past schema matching activities (at the sources already within the system), to successfully match the schemas of new data sources. When self maintaining, the system can learn from the behaviors of the sources to detect failures. Suppose the system knows that two specific sources behave similarly, in that they return very similar answers to the same query (e.g., analogous to the way *amazon.com* and *barnsandnobles.com* behave with respect to book listings). Suppose further that one source starts to behave in a manner highly dissimilar to the other, then the system can predict that a source failure has happened. The research plan (Section 3) gives examples of other learning scenarios.

A very interesting type of entities that the system can learn from is the *multitude of users* who use the current or other systems in the domain. Indeed, whenever the system employs automatic techniques to arrive at a prediction (e.g., "this URL contains a query interface" or "the wrapper no longer extract data for price correctly"), it can ask the users to judge the correctness of the prediction. This way, the enormous burden of the system builder will be spread thinly over a mass of users.

Mass collaboration has been employed quite successfully in open-source software (e.g., *Linux*), product reviews (e.g., *amazon.com*), and collaborative filtering [129]. It has recently attracted the attention of database and data mining researchers. Raghu Ramakrishnan proposed to use mass collaboration to build tech support websites [127], while Rakesh Agrawal and Pedro Domingos applied it to manage user trusts in collaborative environments [4]. In this research, I propose to consider applying it in conjunction with automatic techniques to build data integration systems.

## 1.3 Objectives, Feasibility, and Significance

The goal of this proposal is to make fundamental contributions toward realizing the vision, drawing from the above core idea. Specifically, I will make contributions to the following central challenges:

**System Creation & Evolution:** When constructing the initial system as well as when evolving it, we have to perform a set of labor intensive tasks. *How can we develop effective semi-automatic techniques for these tasks, to reduce human labor?* Prior research has not exploited useful information such as learning from past activities and external data. I will develop novel techniques that leverage such information to maximize task accuracy.

**System Maintenance:** When sources change, system components fail. *How can a system detect such failures, with minimal human intervention?* Very little work has been conducted on this topic, with ad hoc solutions. I will develop principled methods that learn from the current system state, the environment, and the behavior of the sources to efficiently detect system failures.

**Mass Collaboration:** *Can the system further reduce the tremendous labor of the system builder by spreading much of it thinly over the mass of users?* Though promising, mass collaboration has not been applied to building data integration systems. I will develop techniques to apply mass collaboration and examine its limitations.

Further, I will integrate the above techniques to build and evaluate evolving and self-managing data integration systems. I will evaluate the systems in the context of several Web domains (book, real-estate, CS department websites) and a real-world organization (the Illinois Fire Service Institute), as discussed in Sections 3.3-4.

I believe this project is *feasible.* First, I have carried out preliminary research to demonstrate the potential of the approach (see Section 2.2). Second, the project can build upon the wealth of research on data integration – in both the database and AI communities – and leverage much recent advances in machine learning.

Third, the project is also much "in sync" with the broader research landscape. I have mentioned that it fits into the broader paradigm of autonomic and recovery-oriented computing systems [81, 80]. Our project can leverage ideas from these related efforts (as shown in Section 3.2), and vice versa. Within the database area, this research is similar in spirit to self-tuning database efforts at Microsoft [79, 23], IBM Almaden [101, 100], and others [142]. The key difference is that self-tuning research optimizes *physical "knobs"* (e.g., memory buffer size) of traditional database systems, whereas I aim to "optimize" the *semantic "knobs"* (e.g., semantic mappings) of data integration systems.

**Intellectual Merit:** This research will have twofold impact. First, it makes fundamental advances in the current state of the art of data integration. Indeed, it represents a *next logical step* for data integration research, which has moved from addressing architectural issues [65, 98, 141] to optimization of query execution [82, 9, 92, 63, 87], and now onto "optimization" of system development, all toward the goal of making such systems widely practical. Second, many problems underlying this research are fundamental to other integration and sharing scenarios (e.g., data warehouses, peer data sharing, Web service composition), as well as to the autonomic computing challenge. Hence, this research has the potential to make substantial advances in those areas.

**Broader Impacts:** The project will facilitate the widespread deployment of data integration systems, thus resulting in more effective information management and access for society. It plays an integral part in educating next-generation professional workers and researchers. I have graduated 2 female Masters students and am working closely with 3 female students and 1 African-American student. This research will enable me to continue the vigorous training of students from underrepresented groups. The research will help integrate data for rural Illinois fire fighters, and train them in access and use. Finally, data and system artifacts from the project will be disseminated broadly in the research community, to enhance the infrastructure for research and education.

## 2 Prior Research Accomplishments

Prior to joining the University of Illinois at Urbana-Champaign (UIUC), for the previous five years I was a member of both the database and AI groups at the University of Washington. There I worked extensively on schema matching, data integration, and machine learning. Recently, I also carried out preliminary research on duplicate matching and mass collaboration, to explore the idea of evolving and self-managing data integration system. This section summarizes the above works and shows how they support the research plan of this proposal.

### 2.1 Data Integration

**Schema Matching:** To translate user queries, a data integration system must know the semantic mappings between the global schema and the source schemas. As a part of my thesis research on learning to map between structured representation of data [37], I developed the LSD (Learning Source Descriptions) system that employs learning techniques to automatically find such mappings [38].

LSD first asks the user (e.g., system builder) to provide the semantic mappings for a small set of data sources, then uses these mappings together with the sources to train a set of learners. Each learner exploits a different type of information either in the source schemas or in their data. Once the learners have been trained, LSD finds semantic mappings for a new data source by applying the learners, then combining their predictions using a meta-learner. To further improve matching accuracy, I extended machine learning techniques so that LSD can incorporate domain constraints as an additional source of knowledge, and developed a novel learner that utilizes the structural information in XML documents. Experiments with LSD on several real-world domains showed high matching accuracy (71-92%) [38].

LSD therefore is distinguished in that it can learn from *past construction efforts* (previous matching in this case) and *domain knowledge* (e.g., integrity constraints supplied by domain experts) to perform schema matching. Furthermore, the multi-learner architecture of LSD is highly extensible in that it can easily incorporate a new learner as it becomes available. Thus if a new learner has been developed for a certain matching application, it can be reused in related ones. LSD therefore also enables a form of *knowledge transfer across applications.*

Research on LSD has resulted in 2 conference papers (WebDB-00 [41] and SIGMOD-01 [38]), 1 invited paper [40], 1 journal paper (Machine Learning Journal [39]), and another journal paper under preparation. The ideas of leveraging past matchings and using a multi-learner architecture pioneered in LSD have since been adopted by several subsequent works [35, 102].

**Query Optimization:** I also conducted substantial research on query optimization for data integration systems. I developed the STREAMER algorithm that efficiently optimizes the execution of a user query, according to a user preference function (e.g., "minimize time to the first tuple" [60, 114, 97, 116, 146]). The technique adapts AI planning methods [73] to effectively handle a broad variety of preference functions. It significantly generalizes several prior solutions [60, 114, 97, 116, 146]. The work has resulted in 1 workshop paper [46], 1 conference paper (ICDE-02 [45]), and 1 journal paper under preparation.

## 2.2 Evolving and Self-Managing Data Integration Systems

**Duplicate Detection:** Recently I have also developed the PROM (Profiler-Based Object Matching) system that performs duplicate detection: deciding if two given relational tuples refer to the same real-world entity. Numerous solutions have been developed for duplicate detection (e.g., [78, 140, 26, 105, 148, 17, 93, 7, 132, 68]). However, PROM shows that all of them can be substantially improved by exploiting additional information that comes from past duplicate detection efforts, from the data in the domain (e.g., from all movie tuples in the Internet Movie Database at *imdb.com*), and from domain experts and users. Research on PROM has resulted in 1 workshop paper [47] and 2 invited papers [48, 149].

**Mass Collaboration:** In the past year I have also explored the MOBS (Mass Collaboration to Build Systems) approach to building data integration systems. The basic idea underlying MOBS is to ask the *users* of a system to *"pay"* for using it by answering relatively simple questions. Those answers are then used to expand the system. This way the enormous burden of system development is lifted from the system builder and spread "thinly" over a multitude of users. The initial work on MOBS suggested that it is indeed possible to leverage user feedback to build and evolve data integration systems. This work has recently appeared in WebDB-03 [106] and the IJCAI-03 workshop on data integration [52].

## 2.3 Machine Learning & AI

Prior to my Ph.D. research in databases and data integration, I worked extensively in the AI areas of planning, reasoning under uncertainty (focusing on probabilistic methods), and medical diagnostics [74, 36, 42, 72, 70, 75, 44, 43, 85]. In recent years, I have done significant work in machine learning and ontology matching.

**Ontology Matching:** I have built on LSD to develop GLUE, a system that matches ontologies [50], with potential applications in knowledge-base construction and the Semantic Web. In developing GLUE, I formulated a general and efficient learning framework that employs relaxation labeling to exploit a broad range of domain constraints and common heuristics [50, 37]. The framework has well-founded probabilistic interpretations. This is in contrast to previous works that exploited only specific types of constraints and heuristics, in an ad-hoc manner. Research on GLUE has resulted in 1 conference paper (WWW-02 [50]), 1 invited paper [51], and 1 journal paper (VLDBJ-03 [49]).

## 2.4 Summary

My prior research in schema matching, duplicate matching, and mass collaboration suggests that it is possible to build data integration systems that learn to evolve and self manage, and that such systems hold great promise. The challenge of this proposal is then to realize the full potential of the approach. I believe that I am well qualified to tackle this challenge, given my extensive work on this topic.

In particular, my experience in data integration and machine learning gives me a unique background to approach my research goals: I will extend my work in data integration to address the problem of building evolving and self-managing data integration systems, and will leverage my machine learning background to combine learning and database techniques to address the challenges.

## 3 Research Plan

I now elaborate on the research thrusts which have been motivated in the introduction. I describe techniques to create, evolve, and maintain a data integration system effectively, then discuss augmenting them with mass collaboration.

### 3.1 System Creation & Evolution

When creating an initial data integration system, the system builder must execute a set of labor-intensive tasks such as wrapper creation and schema matching (Section 1.1). When the system *evolves* by incorporating new data sources, it must perform the above tasks repeatedly, once per each new source. Hence, to build systems that evolve effectively, a key challenge is to automate these labor-intensive tasks. Numerous works have been conducted on automating several tasks, such as wrapper creation [90, 8, 32, 58, 28], schema matching (e.g., [110, 121, 108, 103, 19, 107, 119, 111, 120, 99, 11, 12, 115, 57], see also [126, 10] for surveys), and duplicate detection [140, 26, 105, 148, 17, 93, 7, 132, 68, 78, 64, 128]; but very few works on others, such as source discovery [20], query interface interpretation [77], and global schema construction.

I plan to improve upon this current status in two ways. First, I will attack a few important and well-studied tasks, specifically *schema matching* and *duplicate detection*, but introduce novel techniques that leverage many types of information in the environment to significantly improve task accuracy. Improving accuracy is critical because such improvement translates directly into reduction in human involvement [108, 39, 34]. Thus, to minimize human intervention while evolving, a data integration system must automate the tasks with as accurate results as possible. Second, I will attack *global schema construction*, an important problem [27, 95] that has not received much attention. In what follows I describe my plan for the above three tasks, then discuss the research's potential for long-term impacts.

#### 3.1.1 Schema Matching

Traditionally, when given two schemas, prior work examines the syntactic clues associated with only those two schemas (e.g., attribute names, types, and their data instances if any) to infer semantic mappings. In many contexts, however, we may have information beyond just the two schemas, such as past matching activities and external data, which can be exploited to further improve matching accuracy.

To illustrate, consider matching source schema *homes.com* with the global schema $G$ in Figure 1. Suppose that the data integration system is checking whether attribute contact of $G$ matches agent-phone of *homes.com*. Since the system does not have any information associated with contact other than its name (recall that the global schema is a *virtual* schema over the data sources), it is uncertain about the match. Now, suppose the system builder has matched $G$ with source schema *realestate.com*, and in particular has matched contact with agent-voicemail of *realestate.com*. Now the system knows more about contact, since it knows that data instances of agent-voicemail are also instances of contact. It can compare these instances with those of agent-phone of *homes.com*, to conclude that they look similar, and hence contact matches agent-phone. This idea led to my work on the LSD system [38, 39] (briefly described in Section 2). In the data integration context, LSD learns from *previous matching* of a set of source schemas into a global schema, to propose mappings with high accuracy for subsequent source schemas.

Several recent works [59, 102] have extended LSD by exploiting also *external domain data* in form of *an external corpus of schemas and some mappings among them.* This scenario arises, for example, when we try to exploit the schemas of numerous real-estate sources on the Web, to help in matching the current source schema, or when we try to create a global schema (as discussed below). To motivate the idea, consider matching attribute A of a schema $S_1$ with B of schema $S_2$. If we know that A matches C of $S_3$ and C matches D of $S_4$, then we can leverage all information at C and D to *know more* about A. Similarly, we can leverage mappings associated with B to know more about B. More information about both A and B should increase our accuracy in predicting whether they match. Other works [35, 12, 77] have also explored variations of this idea, with promising results.

Thus, the idea of exploiting a corpus of external schemas to improve schema matching has great potential. (Indeed, a recent work [59] shows how such a schema corpus can also assist in designing schemas and querying heterogeneous data.) The key challenge is how to effectively exploit the corpus. As the above example with attributes A and B shows, if we know the mappings among the schemas in the corpus, then exploiting it is relatively easy. In practice, however, we usually do not know these mappings, partly because they are very expensive to create. For this setting, no satisfactory solution has been found. The work in [102] attempts to exploit the corpus without creating the mappings, with limited accuracy. The work in [77] addresses this problem by creating mappings among all pair of schemas, but considers only a restrictive setting, where an attribute of a schema cannot match with two or more attributes of another schema.

I propose to develop a general solution to this problem. The solution uses a learning-based matching tool (I will start with LSD) to map all pair of schemas in the corpus, learns from the mappings, uses what it learns to remap all pairs again, and so on, until the mappings converge. The intuition behind this solution is that as long as the matching tool is reasonably good (and many current learning-based tools are, see [37]), the majority of the created mappings will be approximately correct and will provide enough *statistical* information so that we can *learn more* about the attributes. The learned information can be leveraged in the next matching iteration to yield better mappings, and so on. As such, the solution is similar in spirit to many recent AI techniques that learn from unlabeled text data (e.g., [118, 150, 117]). I will explore this connection to harvest and adapt ideas from the related AI techniques for the schema matching context.

### 3.1.2 Duplicate Detection

Duplicate detection is the problem of deciding if two given objects such as relational tuples *match*: that is, if they refer to the same real-world entity. This problem is especially critical in settings where a data integration query touches a large number of sources, since then the number of duplicate tuples can explode and render the query result practically useless to the user. In the data integration context, tuples to be matched usually have overlapping, but disjoint attributes, such as those that come from relational tables with schemas (age,name) and (name,salary). This scenario arises because data sources are typically developed in an *independent* fashion, and therefore are likely to have overlapping, but different schemas.

In this scenario, all prior solutions match tuples *based only on the overlapping attributes* [47]. For example, given the two tuples (9,"Mike Smith") and ("Mike Smith",200K), the solutions would declare a match, based on the names. However, a match here would result in a "Mike Smith" who is 9 year old and has a salary of 200K. This appears unlikely, based on our knowledge, specifically, on the *"profile"* that we have about what constitutes a typical person. This profile tells us that such relationship between age and salary is unlikely to exist. Thus, as human we can easily tell that the above two tuples are unlikely to match.

This idea led to my preliminary work on the PROM algorithm [47], which exploits the correlation among *disjoint* attributes to improve matching accuracy. In the above example, PROM begins by matching the tuples based on the shared attribute name. Then it applies a set of *profilers*, each of which contains some knowledge about what constitutes a typical person. The profilers examine the tuple pair to see if it can plausibly make up a person. A profiler may have knowledge about age/salary correlation, and predict that because the age is 9 and the salary is 200K, the tuples do not make up a person and thus do not match. PROM combines the profiler's predictions to arrive at a final prediction for the tuple pair. Profilers can be manually specified by domain experts, learned from training data, transferred from other matching tasks, or constructed from external data. Thus, the PROM approach is distinguished in that it not only can exploit disjoint attributes to improve matching accuracy, but can also reuse knowledge from previous duplicate detection tasks.

In the work [47] my students and I implemented a preliminary version of PROM, and showed that it substantially outperformed existing methods in two experimental domains. Thus, I believe the PROM approach has great potential for significantly improving the accuracy of duplicate detection. However, significant challenges remain that must be addressed to realize the full potential of the approach. I propose to continue the PROM work and address these challenges.

The first challenge is how to create the profilers. In [47] we asked domain experts to hard code them, and applied several learning methods to create profilers from training data (i.e., from a set of tuple pairs that have been manually matched). I propose to systematically examine the different types of possible profilers and develop methods to create them. I am especially interested in profilers that can be created from external structured domain data (e.g., movie and review profilers from the Internet Movie Database *imdb.com*). The second challenge is how to combine them. In [47] we use a simple method of weighted combination with the weight set manually, in a laborious process. I propose to examine a systematic solution to combining the profilers. Other important challenges include "can we construct profilers from text?" and "can we exploit profilers to also reduce the runtime of duplication detection?". I propose to investigate these challenges in depth.

### 3.1.3 Global Schema Construction

The problem of global schema construction has been identified as critical for data integration [27, 95], but has received very little attention. I propose to consider three common problem cases. The first case constructs a global schema from a set of source schemas. Here, I will develop a method that helps the system builder find *frequent* attributes, that is, those that appear in more than $k\%$ of the schemas, for a prespecified $k$. These attributes are good candidates for being included in the global schema. To find them, a solution is to find the mappings among all schema pairs. I will consider applying the iterative technique that I described earlier for schema matching (Section 3.1.1) for this purpose.

In the second case, suppose we have extracted only a set of HTML pages from each source, how can we construct a global schema? This case arises for example when we have a list of 400 real-estate sources (e.g., from *invisibleweb.com*) and want to leverage them to build a "good" global schema, but do not want to go through the labor intensive process of creating all 400 source schemas. To find most frequent attributes. I will consider using an automatic wrapper construction method (e.g., RoadRunner [32]) to create source schemas, then proceed as in the first case. However, currently automatic wrapper construction is not perfect, hence many source schemas are likely to be incorrect. Thus it is likely that we do not find all frequent attributes. To find additional relevant attributes, I will consider applying techniques from the third case, as described below.

In the third case, given only a collection of data in the domain, which includes HTML pages, relational tables, and text, how can we construct a global schema? I propose to develop a technique that starts by asking the system builder to provide some "seed" concepts that he or she wants to include in the global schema, such as house address and price, together with some sample data instances. The technique then examines the data collection to find and propose *related* concepts. For example, it may state that tax rate frequently appears near price (in the data collection), and hence is presumably related. Suppose the builder includes tax rate in the global schema, then the technique will proceed to find other concepts related to tax rate, and so on. For this purpose, I will consider leveraging the concept network of WordNet [109], as well as techniques for discovering networks of concepts from text (e.g., [133, 123]), information extraction (e.g., [62, 112]), and discovering entities and relations from text [148]. I plan to use the technique developed here in the second case, after a set of frequent attributes have been identified and thus can be used as "seed" concepts.

### 3.1.4 Additional Related Work & Long-Term Impacts

Many of the tasks that we face in building data integration systems (and handling heterogeneous data in general [55]) are labor intensive because they reason with the elusive notion of *data semantics*. Hence, the more information we have, the better we understand data semantics, and the better we can execute the tasks. Because of this, the idea of exploiting external information, such as past activities and domain data, to improve task accuracy is extremely promising. Indeed, variants of this idea have long been applied successfully in speech recognition [83] and statistical natural language processing [104] (e.g., in the form of exploiting text corpora). It has recently received significant attention in the AI community (e.g., learning from unlabeled data [118, 150, 117]) and the IR community (e.g., applying language models for IR tasks [124, 91, 151]). But it has not been applied to address information management problems in the database community, except for the few initial works described above. Therefore, my proposed research in this direction – besides its immediate relevance to evolving data integration systems – can also be seen as trying to significantly push the envelope in the area of dealing with semantic heterogeneity, an area that has repeatedly been identified as an important future research direction [3, 55]. My long-term goal is to build on the work in this proposal to develop a principled and practical framework for exploiting domain data and knowledge to handle semantic heterogeneity.

## 3.2 System Maintenance

After constructing a data integration system, we must monitor it for signs of failure due to changes at the sources, and repair the system whenever necessary. Given the dynamic and autonomous nature of data sources (especially on the Internet), failures arise often [89, 96, 27]. Maintaining a data integration system therefore is a labor intensive undertaking. In the long run, its cost will dominate the cost of system ownership [27]. Hence, developing techniques to reduce the maintenance cost is absolutely critical for the widespread deployment of data integration systems in practice.

Surprisingly, very little work has been conducted on this topic (with the exception of [89, 96], see below). Thus, in this research thrust, I aim to make fundamental advances in this area. I will focus on the challenge of developing effective techniques that detect many failures (i.e., high recall) and produce few false positives (high precision). I leave the next challenge of automatically repairing the failures as a possible extension, as time permits.

### 3.2.1 Problem Space Characterization

First I will develop a characterization of the space of possible failures. The field of fault-tolerant computing defines a failure as “any deviation that occurs from the desired or expected behavior of a system” [84]. This definition applies to our setting as well. The space of failures turns out to be surprisingly diverse. The simplest – and most well-known – type of failure arises when the system components that deal with a specific source produce *semantically incorrect data* because of changes at that source. Examples include failures at the wrapper, at semantic mappings, and at mechanisms to handle the source query interface. I call these *single-source failures.* Prior work on system maintenance [89, 96] has dealt only with such failures, focusing on wrapper problems.

Surprisingly, there are many cases where several sources experience no single-source failure, but together cause a *multi-source failure.* For example, consider attribute book-id of source $A$ which lists ISBN numbers (e.g., “0764515470”). Assume that attribute book-isbn of source $B$ shares the same format, and that a common operation is to *join* books across sources $A$ and $B$ based on their ISBN numbers. Now suppose source $A$ changes the data format of book-id by prefixing “ISBN” before the actual numbers. Since the meaning of book-id has not changed and the wrapper still extracts *semantically correct* data, $A$ does not have a single-source failure. However, the join between $A$ and $B$ is no longer correct, resulting in a multi-source failure for the set of sources $\{A, B\}$. This type of failure has not been identified by prior work.

### 3.2.2 Single-Source Failure Detection

In conjunction with studying the problem space, I will develop a solution for detecting single-source failures, since such failures are fairly common. Consider *wrapper failures*, a special case of single-source failures. To detect wrapper failure, prior work [89, 96] extracts a sample of data from the source when the wrapper is known to be working, and computes a *signature* of this data. If the signature of a subsequent data sample extracted at time $t$ differs significantly from this signature (based on some probability model [89] or statistical method [96]), then the wrapper is declared to fail at time $t$.

The above approach works well for many cases, but still suffers from two serious limitations. First, the signature employed is ad-hoc. It is typically a set of *features* such as the number of tuples extracted, the ratio of numeric vs. alphabetical characters for a certain field (e.g., price), and the data formats [89, 96]; but no method was proposed for selecting a good set of features, even though such selections critically affect failure detection accuracy, as [96] shows. Note that the signature comparison method is also ad-hoc. Such ad-hoc measures make it difficult to transfer the approach across application domains.

Second, the above approach produces a high number of false positives [89, 96]. This is a serious problem because false positives require unnecessary and costly human intervention. I believe this limitation arises partly because the approach does not distinguish between *failure-inducing changes* and *normal changes.* The formers cause the wrapper to produce *semantically incorrect* (i.e., garbage) data. For example, when the source moves the location of the field total-cost, the data extraction rule for this field may extract random characters instead of costs. Normal changes on the other hand merely cause the wrapper to extract data in different formats, for example “$35.7” instead of “35.7”, but the extracted data is still *semantically correct.*

I propose the following learning-based solution for detecting source failure, which addresses both limitations. First, I will consider a very large set of data features, which aims to include *all* features that might be useful in failure detection. The set includes both generic and domain-specific features (as may have been suggested by the system admin). Next, for the data source under consideration, I inject changes, that is, "perturb" the source in many ways (see below), carefully noting which ways lead to correct and incorrect source status. I compute a feature vector (i.e., the signature) for each perturbed source, and assign to it a label "correct" or "incorrect", thus in effect obtaining a set of training examples. Finally, I apply learning methods to this training set to learn a model that predicts a source status (as to having a failure or not). I will start by using the *Winnow* learning method [67], because it is specifically designed to handle a large number of features. *Winnow* can also automatically select from the original set of features a smaller set that does a good job at prediction. Selecting a small set of features is important because a large set will make monitoring a data source very expensive.

The proposed solution bears some resemblance to recent work in building robust systems [25], where failure-inducing changes are deliberately injected into a system to test its robustness. The work [18] perturbs a system to learn a model of interaction among its components. In our case, we inject both *failure-inducing* and *normal* changes, in order to learn to distinguish correct system states from incorrect ones.

Normal changes to be injected can be obtained in many ways. The simplest way is to copy from other sources. For example, suppose prices are stored as "$30.75" at source $A$, and as "30.75" at source $B$. We can simulate perturbing $B$ by changing the format of prices at $B$ to be similar to that at $A$. Here, in a sense we are using not just the data of $B$, but also data of $A$ to *teach* the system the many normal formats of price, so that later if $B$ changes price formats, the system does not mistakenly issue a false positive. *This is an example of how the system learns from itself to do a better job at failure detection. Such teaching is not possible in prior works partly because they consider failure detection for isolated sources, not in the data integration context, where many sources are tied together, as we do here.* Failure-inducing changes can be obtained by following some simple rules. For example, moving the total-cost field to a different location in the schema is almost certain to cause the extracted values of this field to be incorrect, thus, inducing a failure. Domain experts and system admins can suggest additional changes, which can then be compiled into a library to be used across systems in the same domain.

In addition to the above learning-based technique, I will also consider exploiting similarities in source behavior for failure detection purpose, as described earlier in Section 1.2.

### 3.2.3 Multi-Source Failure Detection

I will also develop a solution for detecting multi-source failures. In this setting, static probing as described above largely does not apply, due to the exponential number of source combinations. Instead, I will explore runtime monitoring, building on the ideas proposed in [25]. There, the authors consider failure detection for large, dynamic Internet services which comprise many components (e.g., load balancers, web servers, and backend databases). They tag real client requests as they travel through the system, then use data mining to correlate the believed failures and successes of these requests, to determine which component or set of components are likely to be at fault. Translating to our setting, the client requests are user queries, and the intuition is that if a set of sources "fail", queries involving those sources will be unsuccessful, in that they produce abnormal results (e.g., empty or too many tuples). Note that such runtime techniques are applicable to the single-source failure setting as well.

## 3.3 Mass Collaboration: Leveraging Users for System Development

In the previous research thrusts I have proposed semi-automatic techniques to significantly reduce the workload of the system builder. In this section, I propose a conceptually novel technique to further drastically reduce this workload. The idea is to leverage the *mass of users* – instead of the *system builder* – to help verify and correct the predictions made by automatic methods.

To illustrate, consider the problem of source discovery, which commonly arises when attempting to build large-scale data integration systems on the Web [27, 20]. When a source-discovery tool predicts that a certain HTML page contains a query interface (to a database), instead of asking the *system builder* to verify, the tool can ask the *users* if the interface is indeed a query interface (as apposed to a survey or subscription one). If enough users say "yes", then the tool may conclude that its prediction is correct with high probability. As another example, when a tool interprets a query interface by matching attribute area of the interface with attribute address of a global schema, it can also asks the users to help verify that the matching is correct.

Mass collaboration can potentially be applied in many settings, including enterprise intranets, scientific domains, and the Web. For example, within an organization, the employees can collaboratively build systems that integrate organizational data. Bioinformatists can collaboratively build data integrations system over the hundreds of online bioinformatics sources. Students in a database course can be asked to answer questions when they access the course homepage, as a form of "payment" for the homepage service. The "payments" can then be leveraged to build a data integration system in the CS domain (or to mark up data on the CS department website). As yet another example, imagine that Google asks each user of its service to answer 1-2 simple questions per day, as a form of "payment". It is unlikely that this will make many users defect from Google. In fact, very soon Google can leverage the answers of millions of users to mark up data on parts of the Web and to build data integration systems, thus offering an ever improving service, and attracting even more users.

As described, mass collaboration has the potential to dramatically reduce the cost of building data integration systems and speed their deployment in many domains. But can it be done? In recent works [106, 52] my students and I implemented a preliminary mass collaboration algorithm to find 1-1 semantic mappings for data integration systems. In experiments with both synthetic and real users in a simplified book domain, we showed that (a) the algorithm successfully combined user answers to match schemas, even when there were many malicious users, and (b) real users can handle the cognitive load of answering questions in the book domain.

I propose to extend the above preliminary works significantly, to (1) develop a comprehensive and effective framework for applying mass collaboration to data integration, (2) demonstrate the feasibility of the approach by showing that it can be done in several real-world settings, and (3) evaluate the approach further and explore its limitation. In what follows I elaborate on the above research plan.

### 3.3.1 Develop a Mass Collaboration Framework for Data Integration

**Algorithm:** Users tend to be impatient and do not want to be forced to do cognitively difficult things. Thus, to apply mass collaboration to a task such as source discovery, I propose to first break the task down to a series of simple Boolean statements, then solicit user answers ("yes" or "no") for each statement until the "combined answer" for that statement has "converged" to a value with high probability. For example, suppose that an HTML page contains three form interface $I_1, I_2$, and $I_3$, and that we want to know which one is a query interface, if any. Then we can simply break this problem down into three statements of the form "is form $I_k$ a query interface?" and solicit user answers for each of them.

I note that many tasks for data integration can be easily broken down into questions, such as query interface discovery as described above, query interface interpretation, and wrapper construction. I will systematically consider the tasks that arise during system development and determine which ones exhibit this property. (In fact, some of the above tasks appear very well-suited for mass collaboration. For example, query interfaces are designed *specifically for human consumption*. Thus, users would have little difficulties recognizing a query interface.)

I now briefly propose solutions to the challenges that must be addressed in order to make the above algorithm practical.

**Enticing Users to Participate:** We can force users of the current system to provide answers. Every time a user asks a query, we make him or her "jump through a hoop". The hoop is a dialog box with a simple question to which the user answers by clicking an "yes", "no", or "not sure" button. One can 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 maintain and expand the system. We can also leverage users from a different application, as demonstrated in the CS department example described earlier. Note that such users can also be used to build the initial version of a data integration system.

**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 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 answers of a particular user. The weight is computed from user answers to questions whose answers we already know (these questions can come from a few "training" data sources). The "known" questions are randomly injected.

**Combining User Answers:** Periodically the system will combine user answers to arrive at new values for statements. The combination will use the user weights. *In essence, we can treat each user as a learner, that is, a classifier that has been trained and is ready to make predictions on the system data and attributes.* This immediately suggests the applicability of numerous schemes to combine learners' predictions, as described in several recent works [38, 50, 35, 102]. It also raises the issue of how to learn the accuracy of learners and combine a very large number (e.g., thousands) of learners in an efficient and accurate way. This interesting issue has not been considered in machine learning research.

In [106, 52] I discuss the above challenges in more detail, and briefly propose solutions to additional challenges that arise in applying mass collaboration.

### 3.3.2 Feasibility Demonstration & Further Evaluation

I propose to build several applications to evaluate the feasibility of mass collaboration. I will first describe the applications, then the user populations.

The first application integrates query-able sources in the book and real-estate domains. I will start by soliciting user answers to discover the query interfaces of the sources, then to match these interfaces with a predefined global schema. I will consider each task in turn, and examine if I can develop a mass collaboration solution for it (by being able to break it into a series of simple questions).

The second application that I build will be a data integration system over the websites of the CS departments in the US. Specifically, the system integrates the "professor hubs" of the CS departments. A professor hub is a page that links to the homepages of all professors. The hubs in the system are linked to a query field called professor-homepage, and thus allowing users to search for only homepages that contain a given keyword. The system will also integrates several other types of hubs, including "student hubs", "course hubs", and "research hubs". I will use a learning algorithm (based on the techniques described in [31]) to find likely hubs, then solicit user answers to identify the correct ones.

This is thus an example of building data integration systems on the Surface Web (as opposed to the Deep Web of the first application). My conjecture is that building data integration systems on the Surface Web is well suited for mass collaboration, partly because pages on the Surface Web are designed specifically for human consumption. Thus, a CS user can decide very quickly if a page is a professor hub or not. To test this conjecture, I select the above CS application.

For the above two applications, I will experiment with the following three populations. The first user population will be the 120-150 students who will take my CS 311 course, as well as other courses (e.g., CS 497AD, see Section 6). When a student accesses the course homepage, the student will be asked to answer a simple question. The second user population will be the immediate friends and relatives of members of our research group. And the third user population will be members of our group themselves. These populations thus differ greatly in technical sophistication as well as motivations to participate in the mass collaboration efforts.

If mass collaboration works successfully for the above applications, and if time still permits, I will consider evaluating it in other applications. The first candidate will be the fire fighting domain that I describe in the next section. The second candidate can be building a mass collaboration-enabled data integration system in the bioinformatics domain, based on the current system ENTREZ [144], then advertising to entice bioinformatists to use it.

### 3.3.3 Related Work & Long-Term Impacts

As far as I know, this is the first work to apply mass collaboration to build computing systems. Mass collaboration has been studied for many applications, including building tech support websites [127], knowledge bases [131], user trust networks [4], word sense disambiguation [1, 2], and error correction on CiteSeer [94]. These works ask users to contribute facts and rules in some specified language. This poses a serious problem because potentially *any* fact or rule being contributed constitutes a statement whose validity must be checked. Thus, the number of statements can be very high (potentially in the millions) and checking them can be very difficult in practice. In contrast, the number of statements in a data integration setting is comparatively much smaller and thus potentially much more manageable. As such, I believe building data integration systems is a "killer app" for mass collaboration. If successful, this work will have impact far beyond data integration context. Mass collaboration may also be an effective approach to help build the Semantic Web [13], which proposes to mark up data for efficient and expressive querying. All marking-up-data

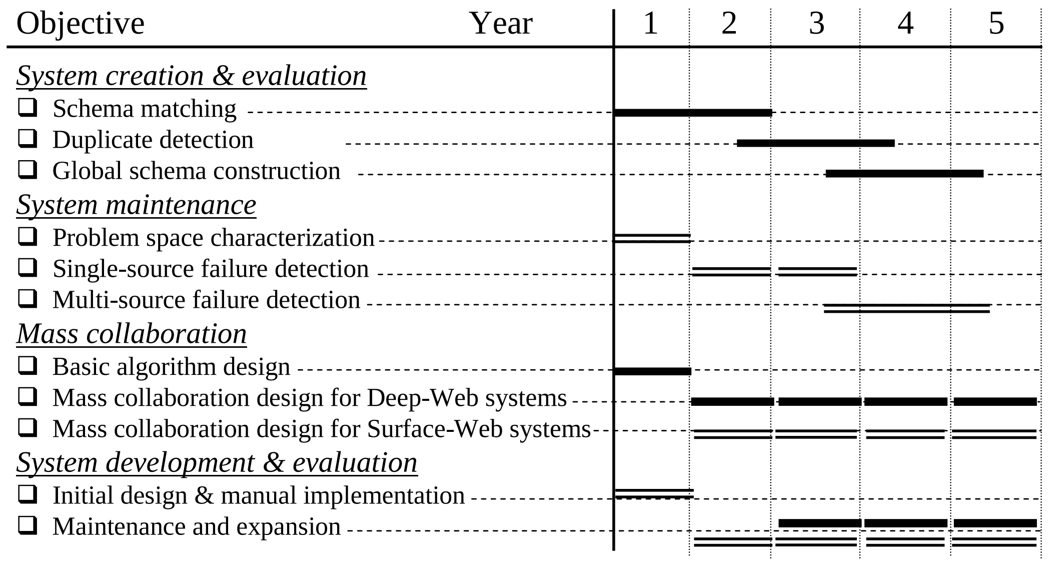

Figure 2: The time line of my research plan: the solid lines represent RA 1, and the other lines RA 2.

solutions so far have assumed the *producers* of a Web page should mark up the page. I propose to consider also the solution that the *consumers* of a page should help mark up the page. For example, users of a CS website (e.g., students) should help mark up the site.

## 4 System Development, Testbeds, & Evaluation

In conjunction with the previous research thrusts, I will develop testbed systems to evaluate and guide the research. Ideally, the testbed suite should be *broadly diverse* in the following important aspects: data characteristics, user population, degree of difficulty for integration, and frequency of changes. Based on these requirements, I plan to use three testbeds: (1) integrating relatively simple query-able Web sources for the book and real-estate domains (sources will be obtained from *invisibleweb.com* and the Deep Web source repository [21]), (2) integrating CS department websites (as described in Section 3.3.2), and (3) integrating the internal databases of the Illinois Fire Service Institute and selected Web sources (see attached letter of support). The Institute databases catalog the entire collection of the Institute and have diverse formats that describe books, multimedia materials, and equipments; the Web sources are those with fire fighting related materials. A typical query that fire fighters would ask is “find me all materials that Midwest fire fighter stations use to train their personnel in handling bio-hazardous situations”.

The systems will be developed in parallel with research in other thrusts. Initially, we will build some small systems manually and some with the help of mass collaboration (see Section 3.3.2). As we develop the techniques in other thrusts, we will use them to evolve and maintain the systems.

The systems and testbeds will be released for public use, and as a research vehicle for the community. In particular, my new course CS497 will benefit from the close interaction with this research, as Section 6 will discuss.

Research solutions will be evaluated primarily on three aspects: accuracy, performance, and reduction in human labor. Evaluating the first two is well understood and should not pose problems. Methods to evaluate reduction in human labor (due to a certain technique) have been developed recently for certain tasks (e.g., see [108, 35, 34] for human labor on schema matching). I will start by adapting these methods to our context. I will also measure the average time spent on developing and maintaining the systems, along the lines of [27], and conduct controlled and blind experiments with volunteers, in which I measure their effort spent on certain tasks.

## 5 Management Plan, Milestones, and Required Resources

Based on my research agenda discussed above (Sections 3), I plan the time line charted in Figure 2. I adopt a phased approach, so that during each year I will demonstrate measurable progress and achieve the specified objectives. The research assumes support for two Ph.D. students working under my supervision. Figure 2 shows that the first student (solid lines) will focus on system creation and evolution and the second student will investigate system maintenance. Both students will participate in developing mass collaboration and building systems. For system development, we also request two server-class PCs.

## 6 Education Plan

### 6.1 Undergraduate Education

**Integration with Research and Real-World Problems:** In the coming years I will be teaching CS 311, our undergraduate database course. To *integrate research with education*, I plan to revamp the course curriculum to add materials on data integration, and to engage the class on the research in this proposal. At the beginning, I will introduce the history of the database field, sketch the current research landscape, and briefly motivate the need for data integration. Next, I will describe our research on evolving and self-managing data integration systems, and ask students to participate via the mass collaboration mechanism placed on the top of the course homepage (as Section 3.3 discussed). Student answers will be used to build several data integration systems and to mark up data on the UIUC CS website, as described in Section 3.3. The students will see the systems *evolve* over time, as the result of their answers. I believe this will be an exciting component of the course which motivates students to learn, and will also be a valuable part of my research. Toward the end of the course, I will give several lectures on data integration, to round out the materials on traditional database systems. Students with strong interests in data integration will be encouraged to explore further research opportunities in my group.

To ground the course in additional real-world issues, I plan a semester-long project that students can do in groups. Each group will select a real-world problem for which they build a Web-based database application in an end-to-end process. I'm currently working with the Illinois Fire Service Institute and the National Center for Supercomputing Applications (NCSA) for database management problems. Students are also encouraged to propose project ideas and problems for consideration.

**Broadening Database Education:** An important goal of my undergraduate education plan is to *train students for data management challenges beyond traditional database settings.* This training makes them better prepared to enter the work force in our Internet world. Data integration is certainly one of the challenges, as discussed above. Going beyond that, I have been working hard with my UIUC database colleagues to revamp the entire UIUC data management curriculum. We introduced two new elective undergraduate courses: CS 310 (Information and Text Retrieval) and CS 312 (Data Mining). These two courses together with CS 311 on Database Systems will offer a well-rounded and solid education in data management. To make the transition and encourage students to take the courses, I plan to invite Chengxiang Zhai and Jiawei Han (my two colleagues working on information retrieval and data mining) to give several lectures in CS 311, to introduce the areas.

### 6.2 Graduate Education

**Beyond Research - Teaching, Networking, & Communication Skills:** Ph.D. education in our profession has usually been somewhat lopsided, in that we train our students vigorously in research, but much less so in teaching, advising, networking, and communication skills. These skills are generally acknowledged as essential for a productive academic/research career. Hence, besides research training, I will take extra steps to ensure that students learn the above skills.

First, I will introduce a *visiting graduate student program.* We researchers often invite one another to visit and give talk. Such visits are highly valuable in that they foster collaboration, knowledge sharing, networking, and in general introduce fresh perspectives. Our graduate students can greatly benefit from similar activities at their level. In Fall 2003, Dr. Chris Clifton at Purdue University and I will start the following program between UIUC and Purdue: a graduate student in the UIUC database group will play the host and invite another student in the Purdue database group to visit for a day, to meet UIUC students and faculty members, and give a talk. Then a UIUC student will visit Purdue, and so on. The visiting student bears the cost of driving, while the host covers the meal expenses. If the program proves successful, we plan to expand it to other nearby universities.

As the second step, I will encourage my graduate students to co-teach a course with me toward the end of their Ph.D. program, and to co-advise a junior graduate student. I have heard from several senior graduate students that such activities gave them invaluable experience in teaching and advising, and at the same time helped them to decide how much they love teaching and advising (and thus an academic career). I myself went through the same experience, and found it to be extremely valuable in making my career decision.

**Underrepresented Group Education:** Another important aspect of my graduate education plan is an emphasis on *training students from underrepresented groups.* I strongly believe that these students are often the ones who return to their roots, to make contributions to their communities, and to serve as role models for others. Hence it is very important that we train strong students from these groups. In June 2003 I have graduated two female Masters students, Arathi Ravi with a job in New York City, and Sarah Sirajuddin with a job at IBM Almaden Research Center. I am currently working with 7 students, including 3 female students (2 Ph.D. and 1 Masters), and 1 African American Masters student.

**A New Course:** As our research community witnesses in general, and this proposal demonstrates in particular, many current data management problems require an innovative blend of database and AI techniques. I believe this trend will only accelerate. Hence, to train students for the new situation, I will design and teach a new graduate course, CS 497AD, in Fall 2004. The first half of the course will be lectures and homework on selected fundamental database and AI techniques. I will focus specifically on machine learning, but will also touch on the comparison between database data models and AI representation languages, and query optimization and AI planning, drawing on my previous research [74, 36, 42, 72, 70]. In the second half, we will study selected data management problems that can benefit from both database and AI techniques. I will ground the course in data integration issues, but additional topics to consider include peer data management, semantic integration, Web mining, and website management. We will examine whether and how each community has addressed the problem, and how the proposed solutions can be combined to obtain a potentially much better one. This half of the course will consist of paper presentations and discussions. There will be a project that can be done in groups of 2-4 students. A preliminary version of this course was offered in Spring 2003, and received enthusiastic comments from the students. For the new version, I will incorporate the infrastructure (e.g., systems and data) developed from this proposed research into the course content, by using it to illuminate discussions and start student projects.

### 6.3 Collaboration and Outreach Activities

I am currently collaborating with Drs. Len Seligman and Arnon Rosenthal at MITRE on a project that develops an open-source industrial strength schema matching system. One of my students is funded by MITRE for working on schema matching (and several others are exploring internship opportunities at MITRE). I expect that this project will make a significant contribution to the research community (I regularly received requests for open-source schema matching tools for research purposes; no such system is available today). Further, the project can provide effective tools for my work on global schema creation, as described in Section 3.1. MITRE has also agreed to advise me on real-world aspects concerning this proposed research program, and to evaluate the techniques as they become available (see attached letter of support).

As mentioned in Section 4, I will assist the Illinois Fire Service Institute in developing a data integration system to provide effective information access to rural Illinois fire fighters. The Institute plans to hold educational seminars across Illinois to train fire fighters in accessing the Internet and using the data integration system. In collaboration with the Institute, we plan to add an educational component on data integration and on how fire fighters can assist in improving the service. Example of assistance include comments, suggestions of new features and data sources, and feedback within the mass collaboration framework. I will assist in training Institute personnel on data integration and system aspects.

## 7 Results of Prior NSF Support

I have not previously been supported by NSF.