# A Platform for Personal Information Management and Integration

Xin (Luna) Dong

Doctoral Thesis Proposal
Department of Computer Science and Engineering
University of Washington

**Abstract**

The explosion of the amount of information available in digital form has made search a hot research topic for the Information Management Community. While most of the research on search is focused on the WWW, individual computer users have developed their own vast collections of data on their desktops, and these collections are in critical need for good search tools.

We study the Personal Information Management (PIM) problem from the data management point of view. A PIM system faces many challenges. First, it should be able to support information search and browsing at the granularity of objects and associations. Second, it should allow automatic population of data instances, and support automatic data cleaning and data integration. Finally, to target life-long personal information management, a PIM system should be resilient to domain evolution and domain obscurity.

We argue that the key for building a successful PIM system is to provide a logical view of one's personal information, consisting of semantically meaningful objects and associations. The thesis of this research is to build a prototype of a PIM system based upon this logical view, and demonstrate how we can leverage such a view to address the aforementioned PIM challenges.

## 1 Introduction

The explosion of information available in digital form has made search a hot research topic for the Information Management Community. While most of the research on search is focused on the WWW, individual computer users have developed their own vast collections of data on their desktops, and these collections are in critical need for good search and query tools. The problem is exacerbated by the proliferation of varied electronic devices (laptops, PDAs, cellphones) that are at our disposal, which often hold subsets or variations of our data. In fact, several recent venues have noted Personal Information Management (PIM) as an area of growing interest to the data management community [1, 56, 35].

As early as 1946, Vannevar Bush described the vision of a Personal Memex [14], which was motivated by the observation that our mind does not think by way of directory hierarchies, but rather by following *associations* between related objects. For example, we may think of a person, emails sent by the person, then jump to think of her papers, and papers they cited, etc. Supporting such associative traversal requires a *logical view* of the objects in one's personal information and the relations between them. However, today's desktops typically store information *by application* and in directory hierarchies. As a result, we need to examine directory structures or open specific applications in order to find information. As an example of the mismatch, information about people is scattered across our emails, address book, text and presentation files. Even answering a simple query, such as finding all of one's co-authors, requires significant work.

We argue that the key for building a successful PIM system is to provide a logical view of one's personal information, based on *meaningful* objects and associations. For example, users can browse their personal information by objects such as Person, Publication and Message and associations such as AuthoredBy, Cites and AttachedTo. With such a logical view, a PIM system can support users in their own habitat, rather than trying to fit their activities into traditional data management. And with such a logical view, PIM services can effectively mine a user's information or activity patterns to provide information more suited for the user, and thus improve the user's productivity in daily life.

Below we give several scenarios to illustrate the challenges a PIM system faces, and briefly show how we can leverage the logical view of personal information to meet the challenges.

### 1.1 Scenarios

**Scenario 1 (Search and browse associated objects).** *Prof. Jones is writing a survey on model management. She wishes her PIM system will help her collect all information on model management, including papers on model management, people working on model management, institutes that conduct model-management research, venues for publishing model-management papers, etc. When Prof. Jones browses the collection of information, she may want to see more details or related information. For example, she may start with a paper and see its authors and citations, and then jump to other publications of one author, etc.* □

**Challenge 1.** *Support information search and browsing at the granularity of objects and associations, independent of the location of the information.*

**Challenge 2.** *Bootstrap the system by automatically populating data instances and removing instance duplications.*

Scenario 1 shows that a person views the world as individual objects connected by various associations. Thus, she often wishes to search information at a fine-grained level and navigate her personal information by following associations, not necessarily being aware of the location of the information. This raises the fundamental requirement for a PIM system, stated as Challenge 1.

Importantly, since users are typically not willing to tolerate any overhead associated with creating additional structure in their personal data, a PIM system needs to populate the data instances mostly automatically. One key problem in this automatic population process is that the same entity in the world might be referred to in many different ways. To truly follow chains of associations and find all the information about a particular individual (or publication, conference, etc.), a PIM system needs to be able to reconcile the many references to the same real-world object. We summarize the above as Challenge 2.

By providing a logical view of personal information, consisting of objects and associations between the objects, we are able to naturally support fine-grained search and browsing, thus addressing challenge 1. Automatic population of such a view is possible because a lot of data formats implicitly entail object instances and associations. For example, an email contains a field for sender and a field for recipients, and a Latex file contains a field for authors. From them we can easily extract instances of classes Message, Person, Article, and associations in types of sendFrom, sendTo, authoredBy. Further, the logical view of personal information highlights the associations between the references and thus provides additional evidence for removing instance duplications.

**Scenario 2 (Integrate external data).** *Prof. Jones wishes to find out which of her acquaintances has published in this year's SIGMOD, and meanwhile store information for papers in her area for*

*future use. She has a webpage with Sigmod accepted papers. However, going through the list is tedious and error-prone, and picking out interesting information for storage (maybe in another format) is even more labor-intensive. A PIM system is expected to do the above for her automatically or semi-automatically.*

*Next time when Prof. Jones gets a webpage with a list of papers, such as VLDB accepted papers, she may want to apply the same task and have her PIM system to do it automatically for her.* □

**Challenge 3.** *Support integration with external data on-the-fly.*

As illustrated in Scenario 2, a user often needs to integrate external data and query across personal and external information. We refer to such integration tasks as *on-the-fly integration*, because they are light-weight tasks performed by individuals for relatively transient goals. In contrast, today's data integration systems typically support heavy-weight tasks for queries that occur very frequently in organizations (*e.g.*, customer relationship management and integrated catalog search). An important challenge for a PIM system is to fundamentally change the cost-benefit equation associated with integrating data sources, and aid non-technical users to easily integrate diverse sources. A logical view of personal information can be used as an *anchor* into which we can integrate external sources.

**Scenario 3 (Personalize the domain model).** *Prof. Jones wishes to extend her personal domain with an object class for* projects*; however, she encounters uncertainty in modeling this class. She wants to capture the association between people and projects: a person can* participate *in a project. But soon she realizes not all participants are equal, and there are various kinds of participation modes. For example, a project may have a programmer, a member in the initial planning phase, advice-giver, etc. She cannot anticipate all the possible participation modes nor classify them precisely. She wishes her PIM system to be tolerant with such uncertainty, and help her refine her domain model in the future.* □

**Challenge 4.** *Be resilient to domain evolution and domain obscurity.*

An ideal PIM system shall allow a user to personalize her domain according to her own view of the world. However, in the process of modeling, it often happens that users are not able to give a precise model, or the domain is inherently evolving (it is especially true for personal information). As we aim at life-long personal information management, the PIM system should be able to capture the evolution of the personal domain. A logical view of personal information provides the flexibility of modeling the domain at different granularities and adapting to domain evolution.

## 1.2 Thesis hypothesis

*The goal of this research is to build a prototype of a PIM system that provides a logical view of one's personal information based on meaningful objects and associations, and to demonstrate how we can leverage this logical view to address the aforementioned PIM challenges.* In particular, we expect the following deliverables of the thesis:

- An approach to automatically build a database of object instances and associations between the instances. In [26, 15, 27] we described how to extract object instances and associations from one's personal data, and proposed a reference-reconciliation algorithm that is well-established for the PIM context. Our future work is to support incremental data collection and reference reconciliation.

- An algorithm for integrating personal data and external data on-the-fly. In [26] we described a preliminary algorithm and we are going to implement and evaluate the algorithm.
- Algorithms of data analysis for supporting several personal-information search-and-browsing modes, aiming to improve the user's productivity in daily life. This is ongoing work.
- The concept of *malleable schema* as a modeling tool that enables incorporating both structured and unstructured data from the very beginning, and evolving one's model as it becomes more structured. This is ongoing work.
- A PIM system that integrates the above components and serves as a platform for other PIM services.

**Organization:** This proposal is organized as follows. Section 2 surveys the state-of-the-art PIM systems. Section 3 overviews our solution, including the data model and the system architecture. Section 4 discusses reference reconciliation; together with Section 3, it addresses Challenge 2. Section 5 proposes on-the-fly integration and addresses Challenge 3. Section 6 states several data analysis techniques and demonstrates how we address Challenge 1. Section 7 discusses the extension of our data model to meet Challenge 4. Finally, Section 8 concludes.

## 2 State-of-the-Art PIM Systems

A number of PIM projects [55, 30, 39, 29, 36, 37, 83, 71, 54, 91, 7] studied ways to effectively organize and search personal information. They all attempt to go beyond the traditional hierarchical directory model and present a unified user interface for personal data. We now survey several main ones and discuss where their data-models fall short.

The Stuff I've Seen (SIS) project [30] and the Google Desktop Search toolkit [41] consider personal information as a collection of documents with *indices* on the text of the documents. They index all types of information (in files, emails, webpages, etc) and emphasize access through a unique full-text keyword-search, which is independent of the application storing the data.

MyLifeBits [39] views personal data as a *network* of documents. Nodes in the graph represent documents and annotation meta-data; edges represent the annotate relationship. MyLifeBits focuses on integrating text and multimedia objects, allowing annotating a file by another file, or by manually adding text annotation or audio annotation.

Placeless Documents [29] models personal information as *overlapping collections* of documents. It annotates documents with property/value pairs, and groups documents into overlapping collections according to the property value. It also enables annotation with executable code in the form of active properties. The Haystack project [78] once explored the same intuition and modeled data as *dynamic hierarchies* of documents. The hierarchy is called *dynamic* because users can use the properties in an arbitrary order to narrow down the search space (in contrast to following a fixed order in the traditional directory model).

The LifeStreams project [36] views personal information as a *sequence* of documents. It organizes documents in chronological order and allows the user to view the documents from different viewpoints in terms of time.

While the above PIM systems all capture some aspect of personal data and provide convenience for information access in certain modes, none of them can satisfy the user needs stated in the three scenarios. The fundamental reason is that they do not support a logical view on one's personal data. In fact, among these PIM systems, only MyLifeBits manages to explicitly capture associations, but it captures associations at a coarse granularity: it distinguishes neither different classes of instances

nor different types of associations. Further, the above PIM projects all consider information at the document level. As a result, they cannot integrate structured data in a semantically meaningful way, or effectively facilitate reference reconciliation. Even reconciling different versions of the same document in these systems is difficult. As for automatic data population, MyLifeBits highly relies on manual annotations, and LifeStreams calls for properties that are largely set by hand, Finally, the above systems are rather fixed than extensible in the sense that users cannot easily apply their knowledge on the domain and extend the data model.

The Haystack project [55] models personal information as objects and associations between objects, and has successfully demonstrated how they can utilize this model for personalized information presentation. Associations in Haystack can be automatically extracted from certain fields of documents, set up by observing the user's behaviors such as browsing trails, or added manually by the user [79]. However, Haystack mainly focuses on rendering of the information, but has not addressed the problems of data integration, data cleaning, data analysis, and model extension, which will be the emphasis of our research.

Finally, the Information Retrieval Community and the Human Computer Interaction Community have conducted research that study how people organize personal files [63, 57, 6, 34], emails [89], and web information (bookmarks) [77, 2, 85, 84, 51]. Boardman and Sasse [12] compared the long-term organization behavior for the above three types of information. These research results lay the foundations for our PIM system design.

# 3 Overview of Our Solution

In this section we overview our PIM system prototype, SEMEX (short for SEMantic EXplorer). First, we describe the data model SEMEX uses. Then, we describe the system architecture of SEMEX.

## 3.1 Data model

SEMEX provides a logical view of one's personal information, described by a domain model. A domain model is close in spirit to an E-R model, containing object *classes* and *associations*. Each object class contains a set of *attributes*, and an attribute value is of atomic type (e.g., string, integer, etc.). Each association has a *domain* and a *range*, both being classes. For convenience, for a particular class $C$, we call the attributes of $C$ and the associations with $C$ as the domain together as the *properties* of $C$. We support class hierarchies and property hierarchies.

A data instance of a domain model is called an *association database*. We can view an association database as a graph, with *object instances* as nodes and *associations* as edges.

## 3.2 System architecture

Figure 1 depicts the components of the SEMEX system. SEMEX has three sub-modules: the domain management module plays the central role by providing and managing the domain model; the data collection module is responsible for data extraction, integration, cleaning and indexing; the data analysis module analyzes data for search and browsing. We now briefly explain each of these modules.

**Domain management module:** The domain management module provides the domain model for the other two modules. SEMEX provides a default domain model to the users; however, one of the

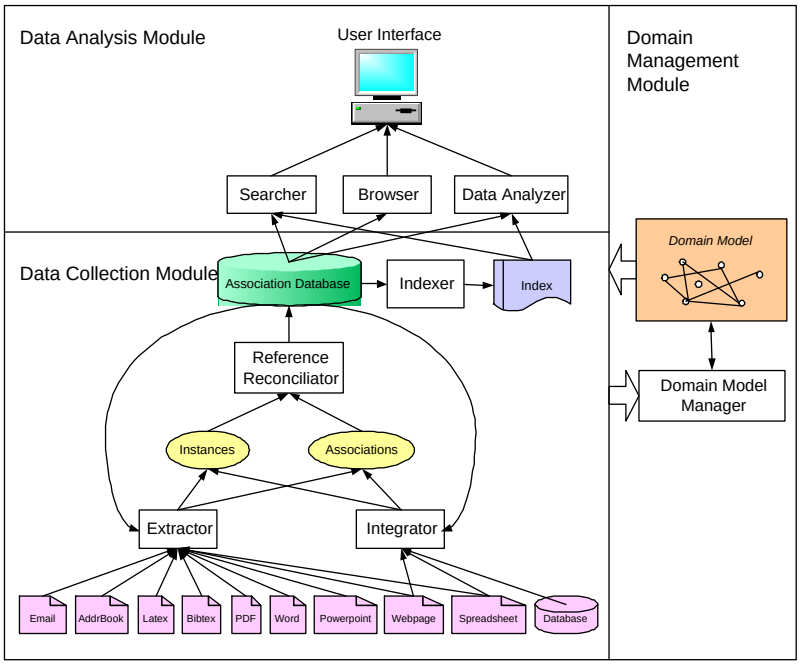


Figure 1: SEMEX architecture. The system is made up of three modules. The *domain management module* is the central part and manages the domain model. The *data collection module* extracts objects and associations from the user's personal information and external data sources, and stores the results in the association database. Users access the association database through the *data analysis module*.

important features of SEMEX is that it allows a user to personalize her domain model. The *domain model manager* component supports model personalization. In addition, it learns from previous integration experiences and browsing experiences to suggest possible domain-model refinement.

**Data collection module:** SEMEX provides access to data stored in multiple applications and sources, such as emails and address book contacts, pages in the user's web cache, documents (e.g., Latex and Bibtex, PDF, Word, and Powerpoint) in the user's personal or shared file directory, and data in more structured sources (e.g., spreadsheets and databases). SEMEX starts data collection by using a set of object-and-association *extractors*. The extracted objects are processed by the *reference reconciliator* to reconcile multiple references to the same real-world object, and the results are stored in the *association database*. Instances in the association database enable more data extraction (such as the association mentionedIn), and facilitate the *integrator* component to integrate external structured data sources. Finally, object instances in the association database are indexed for fast access.

We now briefly state several ways to automatically extract object instances and associations. In many cases, objects and associations are already stored in the data sources and only need to be extracted into the domain model. For example, a contact list already contains several important attributes of persons. Further, a rich set of objects and associations can be extracted by analyzing specific file formats. For example, authors can be extracted from Word documents and Powerpoint presentations. In more complex cases, SEMEX needs to examine data sources of multiple types. For example, citations can be computed from Latex and Bibtex files. Finally, new associations can be computed from existing ones, such as one's co-authors.

**Data analysis module:** SEMEX offers its users an interface that combines intuitive browsing and a variety of query facilities, ranging from simple keyword search to sophisticated association query

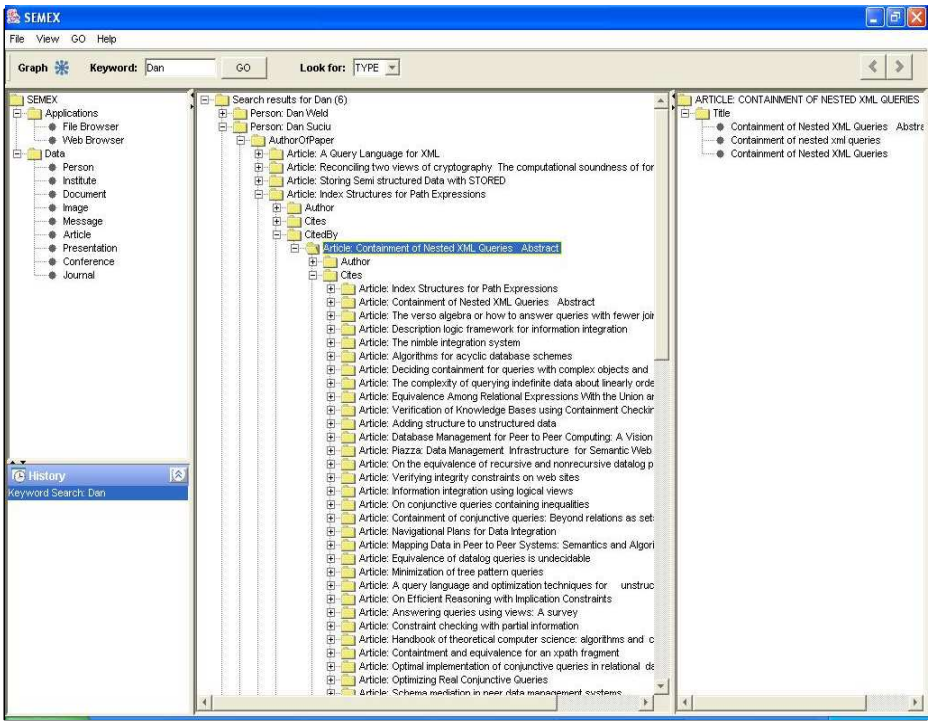


Figure 2: A sample screenshot of the SEMEX interface. The user can formulate a keyword query (top left), a more specific selection query (top middle), or a sophisticated association query (hidden to avoid clutter). SEMEX displays all the information about a particular individual and enables browsing the information by association. As seen in the bottom of the right pane, SEMEX needs to reconcile multiple references to the same real-world object.

(describing an association chain with use of variables) (see Figure 2). The association browsing and search functions are conducted by the *browser* and *searcher* components, respectively. Further, *data analyzer* analyzes people's information and activities, and triggers certain notifications and alarms when an event occurs (for example, when a user opens a document, SEMEX will report the number of the user's acquaintances occurring in the document).

In the rest of the proposal, we address several key technical issues in personal information management, including reference reconciliation, on-the-fly integration, association search and browsing, and domain model evolution. For each of the above issues, we first state the challenges in the PIM context and propose our solution, then compare our work with related work.

# 4 Reference Reconciliation

Since the data SEMEX manages is very heterogeneous and it needs to support multiple sources of associations, it is crucial that the data instances mesh together seamlessly. In data extraction and integration SEMEX generates *references*: each reference partially specifies an instance of a particular class–it has a set of values (possibly empty) for each attribute of that class; and several references may refer to the same real-world object. *The reconciliation algorithm tries to partition the set of references in each class, such that each partition corresponds to a single unique real-world entity, and different partitions refer to different entities.*

As an example, Figure 3(b) shows a set of references extracted from a personal dataset according to the domain model (fragment) shown in Figure 3(a) (for clarity, we denote association properties

Person (name, email, *coAuthor, *emailContact)
Article (title, year, pages, *authoredBy, *publishedIn)
Conference (name, year, location)

**(a) Personal Information Schema**

**Article** $a_1$=({"Distributed query processing in a relational data base system"}, {"169-180"}, $\{p_1, p_2, p_3\}$, $\{c_1\}$)
$a_2$=({"Distributed query processing in a relational data base system"}, {"169-180"}, $\{p_4, p_5, p_6\}$, $\{c_2\}$)

**Person** $p_1$=({"Robert S. Epstein"}, *null*, $\{p_2, p_3\}$, *null*)
$p_2$=({"Michael Stonebraker"}, *null*, $\{p_1, p_3\}$, *null*)
$p_3$=({"Eugene Wong"}, *null*, $\{p_1, p_2\}$, *null*)
$p_4$=({"Epstein, R.S."}, *null*, $\{p_5, p_6\}$, *null*)
$p_5$=({"Stonebraker, M."}, *null*, $\{p_4, p_6\}$, *null*)
$p_6$=({"Wong, E."}, *null*, $\{p_4, p_5\}$, *null*)
$p_7$=({"Eugene Wong"}, {"eugene@berkeley.edu"}, *null*, $\{p_8\}$)
$p_8$=(*null*, {"stonebraker@csail.mit.edu"}, *null*, $\{p_7\}$)
$p_9$=({"mike"}, {"stonebraker@csail.mit.edu"}, *null, null*)

**Conf** $c_1$=({"ACM Conference on Management of Data"}, {"1978"}, {"Austin, Texas"})
$c_2$=({"ACM SIGMOD"}, {"1978"}, *null*)

**(b) Raw References**

$\{\{a_1, a_2\}, \{p_1, p_4\}, \{p_2, p_5, p_8, p_9\}, \{p_3, p_6, p_7\}, \{c_1, c_2\}\}$

**(c) Reconciliation Results**

Figure 3: The (a) schema, (b) references, and (c) reconciliation results.

with "*"). All instances are extracted from two Bibtex items except that $p_7$ to $p_9$ are extracted from emails. Figure 3(c) shows the ideal reconciliation result.

Reference reconciliation is a hard problem in general. It has been the subject of several recent research efforts (see [43, 11] for recent surveys). Most previous works considered techniques for reconciling references to a *single* class. However, the PIM context presents a complex information space: there exist instances of multiple classes and rich relationships between the instances. Though we can apply previous methods to each type of reference in isolation, we miss the rich information carried in the relationships between the instances.

Furthermore, previous techniques assume there are several attributes associated with every reference, and therefore a reasonable amount of information to consider in the reconciliation decision. In the PIM context, this assumption often does not hold. For example, a Person reference often has values for only one or two attributes. In Figure 3, references $p_5$ and $p_8$ do not have any attributes in common.

Finally, most previous techniques assume each attribute has a single value. In the PIM context, some attributes are multi-valued, so the fact that two attribute values are different does not imply that the two references refer to different real-world objects. For example, two Person references with completely different email addresses may refer to the same person. This phenomenon is especially common when we are considering real-world entities that evolve over time.

### 4.1 Key aspects of our solution

In [27] we described a reference reconciliation algorithm that is well suited for the PIM context. The key ideas underlying our approach are the following. First, we make extensive use of *context information* (the associations between references) to provide evidence for reconciliation decisions. In our example, we notice that $p_5$ has co-authored articles with $p_6$, and $p_8$ has email correspondences with $p_7$. If we decide that $p_6$ and $p_7$ are the same person, we obtain additional evidence that may

lead us to reconcile $p_5$ and $p_8$. Second, we *propagate* information between reconciliation decisions for different pairs of references. For example, when we decide to reconcile two papers $a_1$ and $a_2$, we obtain additional evidence for reconciling the conference references $c_1$ and $c_2$. This, in turn, can increase the confidence in reconciling other papers published in $c_1$ and $c_2$. Third, we address the lack of information in each reference by *reference enrichment*. For example, when we reconcile two person references, we gather the different representations of the person's name, collect her different email addresses, and enlarge her list of co-authors and email-contacts. This enriched reference can later be reconciled with other references where we previously lacked information for the reconciliation.

Based on the above intuition, we have developed a reconciliation algorithm that improves over the recall of previous techniques without sacrificing precision. The future work is to design an efficient incremental reconciliation approach, applied when new references are inserted to an already-reconciled dataset.

### 4.2 Related Work

The problem of reference reconciliation, originally defined by Newcombe, et al. [72], was first formalized by Fellegi and Sunter [33]. A large number of approaches that have been since proposed [10, 47, 65, 90, 86] use some variant of the original Felligi-Sunter model. Reconciliation typically proceeds in three steps: first, a vector of similarity scores is computed for individual reference pairs by comparing their attributes; second, based on this vector, each reference pair is classified as either *match* or *non-match*; and finally, a transitive closure is computed over matching pairs to determine the final partition of references. Attribute comparison is done by using either generic string similarity measures (see [21, 11] for a comprehensive comparison and [18, 86, 10] for recent adaptive approaches) or some domain-specific measures (e.g., geographic locality in [5]). The classification of candidate reference pairs into match and non-match pairs is done through a variety of methods: (a) rule-based methods [47, 58, 38, 50] that allow human experts to specify matching rules declaratively; (b) unsupervised learning methods [90] that employ the EM algorithm to estimate the importance of different components of the similarity vector; (c) supervised learning methods [76, 19, 86, 82, 11] that use labeled examples to train a probabilistic model, such as a decision tree, Bayesian network, or SVM, and later apply the model to identify matching pairs; and (d) methods that prune erroneous matching pairs by validating their merged properties using knowledge in secondary sources [24, 28, 67]. The above works all assumed that the references for reconciliation are of a single class.

The idea of capturing the dependencies between reconciliation decisions has recently been explored in [75, 74]. Both approaches entail learning a global detailed probabilistic model from training data, and having the entire reconciliation process guided by that probabilistic model. In complex information spaces that contain multiple classes and complex associations between the classes, learning such a model is hard, if not infeasible. In contrast, our approach provides a mechanism for exploiting influences between reconciliation decisions, and it allows applying different domain-specific models (either heuristic or learned) for particular classes of references. In [9, 52], associations are used to compute similarities and relate reconciliation decisions. Their proposed heuristics are just a subset of the many heuristics we use, and further, we are considering a much more complex domain.

Finally, several works [47, 65, 18, 50] have addressed the computational cost of reference reconciliation. Our algorithm follows the spirit of the canopy mechanism [65] to improve efficiency.

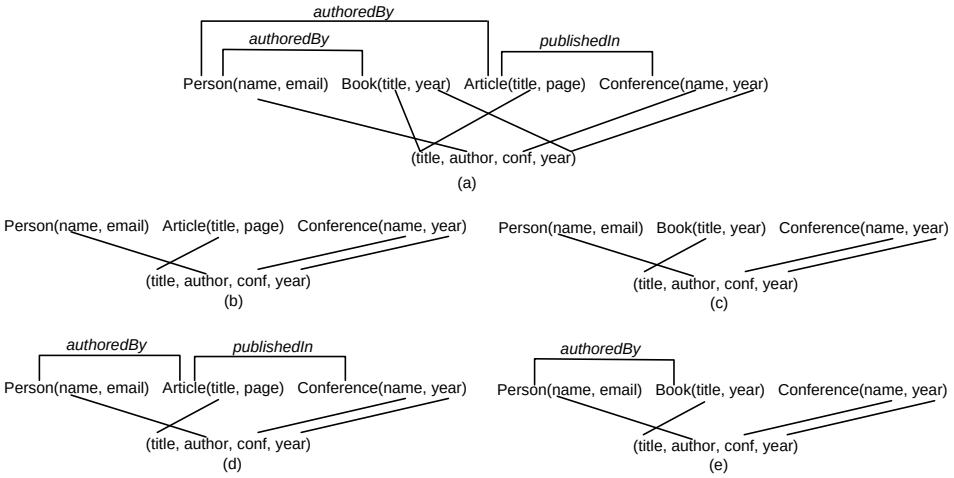


Figure 4: An example schema mapping process. (a) The fragment of SEMEX domain model and candidate correspondences with the columns of the wrapped webpage. (b)(c) Two alternative *matching plans*. (d)(e) The set of objects and associations that will be populated by an item in the wrapped webpage according to the matching plans shown in (b) and (c). The mapping plan in (e) generates *dangling* instances of type Conference, thus is less preferred than the mapping plan in (d).

# 5 On-the-fly Data Integration

One of the main objectives of SEMEX is to leverage the logical view of personal information to facilitate a wide range of *on-the-fly* data integration tasks; that is, to combine information from multiple online sources. The key component in such integration tasks is to establish the semantic relationship between the external data source and the SEMEX domain model. *Formally, it takes as input the domain model and an external schema (relational database schema or XML schema), and generates a set of queries such that evaluating the queries on the external data source will generate a set of class and association instances of the domain model.* In some cases, this step may need a preceding step that guides users to transform the external data into a structured form, such as reading the cells of a spreadsheet or extracting data from a webpage into a relational table. In Scenario 2, the webpage with accepted papers might be wrapped as a table with schema (title, author, conference, year). Now we need to map it into our domain model containing classes Person, Book, Article, Conference, etc.

Previous work divided schema mapping into two separate steps. In the first, often called *schema matching*, we find *candidate* correspondences between the attributes of the two schemas. In the second, referred to as *query discovery* [68], we build on *exact* correspondences and create mapping expressions that specify how to translate data from one schema to another. Note the gap between the output of the first step and the input of the second step. User's input is important to choose the exact correspondences from the candidate ones to fill in the gap.

In the PIM context, there can be many classes, and some attribute names (such as name, title and year) are common across classes. In practice, this implies that the matching phase will yield a large number of candidate correspondences; in our example, the webpage column title can be matched to many attributes called "title". Manually filtering inappropriate candidates is often tedious and requires a good understanding of the domain model, so contradicts the spirit of on-the-fly integration.

Further, previous work assumed the two schemas for mapping are fixed and well established. In the PIM context, however, the integration process may involve *extending* the user's domain model.

When the algorithm cannot find a good mapping, it needs to examine whether it is because the domain model is missing some required classes or associations, and if so, it shall guide the user in extending the domain model.

### 5.1 Key aspects of our solution

In [26], we proposed a preliminary algorithm for matching a relational schema into a domain model. The underlying idea is to exploit the associations for pruning inappropriate matching candidates, rather than requiring users to do it by hand. Specifically, we explore the heuristic that a database tuple seldom corresponds to several real-world objects that are not associated with each other. The logical view we provide treats associations as first-class citizens and enables applying this intuition.

Figure 4 demonstrates the steps of our algorithm. We first generate the matching candidates (see Figure 4(a)). We then enumerate all possible *matching plans*, each of which contains a subset of candidate correspondences (Figure 4(b and c) show two alternatives). For each matching plan, we identify the set of classes and possible associations, and generate the mapping query, called a *mapping plan* (as depicted in Figure 4(d and e)). Now we use a set of common heuristics to order all mapping plans. One important heuristic suggests that a well-established mapping plan is less likely to generate a *dangling* instance; that is, an instance not associated with any other generated instances. By applying this heuristic to our example, we prefer the mapping plan in Figure 4(d) to the one in Figure 4(e). The final result is a ranked list of mapping plans that the user can refine, provide feedback on, or accept. If in the end we cannot find a mapping plan that observes all heuristics, we will consider extending the domain model with associations, classes, or attributes.

The above algorithm addresses mapping a single database table into the domain model. While we need to conduct experiments to verify the algorithm, we also need to extend it for mapping a database with multiple tables, or an XML schema, into a domain model.

### 5.2 Related work

Schema matching has been studied intensively in the last ten years (see [80] for a recent survey). The work in this area has developed several methods that try to capture various clues about the semantics of the schemas. Such clues include element names, descriptions, data types, constraints [60, 8, 62, 53], instances [60, 23, 25], and structures [70, 17, 8, 62, 73, 59, 94, 69, 16, 93, 66, 46, 92] and instances [60, 23, 25]. Some approaches resort to auxiliary sources, such as taxonomies, dictionaries, and thesauri [8, 31]. Among the many sources of evidence considered in our algorithm, we emphasize association as an important semantic clue. When translated into relational terms, associations correspond to referential constraints. However, to the best of our knowledge, there is no work exploring the same heuristics as ours.

Several works have considered how to learn from previous mapping experiences for obtaining better mapping results [23, 22, 61]. In our context, SEMEX is familiar with the domain model and data instances, and it has witnessed a large number of integration activities and might have collected a *corpus* of schemas. We plan to leverage previous results in this area to further improve the accuracy.

Our work is also different from traditional query discovery work [68], which takes exact element correspondences as input. We, instead, find schema mapping based directly on candidate matchings that are automatically generated and error-prone. Our query discovery step effectively prunes inappropriate matching candidates, and generates a ranked list of possible mapping plans.

# 6 Personal Information Search and Analysis

The most important usage of a PIM system is to aid finding information on one's desktop. Consider Scenario 1, and 2. With a logical view of one's personal data, a user can search everything on her desktop related to a keyword. For each returned object instance, she can see its different representations, its associated instances, and also how the instance first occurs in her personal information. When she opens a document or a webpage, the PIM system will help her find all the instances she already knows, and the instances related to the ones lying on her desktop. In this section, we demonstrate several data search modes that can help users improve productivity in their daily lives, and state the challenges. Because of the lack of space, we propose our solution for only one task (the most complex one), namely, *intelligent keyword search*.

**Finding instances related to a keyword (Intelligent keyword search):** Given the logical view of personal information, SEMEX provides a search mechanism that is more intelligent than simple keyword search. Consider Scenario 1, when searching for the keyword "model management", not only will SEMEX return the instances representing papers and presentations that contain "model management" in the text, but also SEMEX will return instances representing people working on this area, conferences for publishing papers on this topic, etc. It is unlikely that "model management" is a name of a person; but a person may have written several papers or given several presentations on model management, or is involved in a model management project and thus has sent and received a lot of emails on model management. Hence, in addition to finding objects whose attribute values contain the given keywords, we need to explore the association network to find *tightly* associated objects. Thus, we will not return a person instance which by chance is a receiver of only a couple of emails on model management. Here the first challenge is to find a meaningful metric for *connection tightness*. The second challenge is to conduct the search efficiently at runtime.

Now consider the ranking of the returned instances. Traditionally we rank them by the TF/IDF measure [81]. However, such a measure cannot precisely capture the importance of an instance (such as the significance of a paper in the model management area), and is often biased by the length of the text. Also, the TF/IDF measure does not apply to instances whose attributes do not contain "model management" in the text.

**Finding association chains:** In daily life a user often tries to remember how she gets to know a person, an article, etc. SEMEX attempts to answer such questions by finding the *association chains* between the instance of interest and the instance representing the user herself. An example association chain is as follows: a person is *mentioned in* an email that is *sent to* the user. In general, we may want to find association chains between arbitrary pairs of instances. Note that the shortest path between two instances is not necessarily the desired one; instead, the path that reflects the first time or the most recent time that the two instances interact is of interest. The challenge is to find such chains without redundant instances on the path.

**Finding patterns in external data:** As illustrated in Scenario 2, when a user is given a new piece of information, such as a document, a webpage, or a spreadsheet, she often starts by finding the object instances she already knows, and then associates them with newly discovered instances and so enlarges her information space. SEMEX facilitates this process by searching a new data source for instances already existing in the association database. A more general requirement is to find similar instances; e.g., finding papers in the same field. The main challenge is to conduct this task *efficiently* and be tolerant of the different representations of an instance.

### 6.1 Intuition for intelligent keyword search

We now focus on the first task and propose a possible solution. A naive way to fulfill this task is to first find all instances whose attribute values contain the required keywords, and then explore the association network for their associated instances. However, traversing the association network is computationally intensive and doing it at run-time can reduce the search efficiency. We propose to index an object on its attributes, and in addition on the attributes of its associated objects. For example, if a person instance with name "Bernstein" has authored a paper on "model management", we index the person instance both on the keyword "Bernstein" and on the keyword "model management". With such indices, when searching "model management", SEMEX will return this person instance as well as the paper itself, and there is no extra burden at runtime.

A refinement of this algorithm treats keywords from the instance's own attributes and keywords from its associated instances differently. For example, when we index a Person instance, 5 occurrences of a keyword in its associated papers are considered as one occurrence, and 100 occurrences of a keyword in its associated emails are considered as one occurrence. This refinement has two advantages. First, it implies a definition for *connection tightness*: when searching "model management", a person instance is returned if she has authored no less than 5 model-management papers, or sent and received no less than 100 emails on this topic, or some combination. Second, this refinement helps in ranking: when applying the TF/IDF measure, we naturally rank a person who published more related papers higher, and rank a person who published 5 papers higher than a person who sent 5 emails on this topic. While such weights can be set manually by applying domain knowledge, how to determine them automatically by observing the association network deserves further study.

Next we consider ranking. Inspired by the Google search engine, we rank objects in a search result by a combination of (1) a *relevance score* computed using the TF/IDF metric, (2) a *usage score* reflecting the creation time, last modification time, and visit frequency of the instance, and (3) a *significance score* that measures the importance of the object in the database. The significance score is obtained in a way similar to the PageRank algorithm [13], using associations as links between objects; however, associations are weighted differently based on their types (e.g., AuthorOf is more important than MentionedIn).

### 6.2 Related work

Our work shares many underlying ideas with web search engines. Given a keyword, a web search engine returns all webpages containing the keyword, ranked by a combination of criteria including the PageRank measure. In addition, some search engines treat link description texts as the description for the page being linked to and use this information for ranking. While we can consider an instance as a page and an association as a hyperlink, our context is much more complex in two aspects. First, instances and associations are of different types and so we often need to treat them differently. Second, in our context there does not exist any explicit link description, so indexing on associated instances becomes tricky.

In the database community, recent works have studied applying keyword search to relational databases [40, 49, 3, 4] or XML databases [88, 45]. Among them, Shanmugasundaram et al. [45] extended the PageRank algorithm for XML documents. Other works adopted the TF/IDF metric and Chaudhuri et al. [4] extended it by applying a similar metric on queries. Our work differs in that we are using a combination of many metrics, geared towards ranking instances in an instance-and-association network. Further, in [40, 3, 49] keyword search considers not only tuples that contain attributes with the given keywords, but also tuples that can join with some other tuple containing

the keywords. We share the same spirit but are expecting a much higher efficiency. Moreover, we use a more delicate measure, connection tightness, to improve the search precision.

# 7 Malleable Schemas

A domain model is a very structured specification of the personal information. It assumes that the user *knows* the structure that she's trying to capture and that it *can* be specified. However, in the modeling process, one or more of the following often hold:

- The structure is inherently evolving over time and by nature there will always be parts of the domain that cannot be precisely modeled.
- The domain is extremely complicated and there is no obvious structure for the domain at certain places.
- The borders between the structured and unstructured parts of the data are fuzzy, and therefore the modeling paradigm needs to support smoother transitions between the two parts.
- The structure of the domain is not completely known at modeling time, and may become clearer as the user sees more data instances.
- Trying to model every detail of items found on one's desktop would be too overwhelming for a typical user, and maintaining the model would also be impractical.

To address these problems, we need a modeling tool that enables incorporating both structured and unstructured data from the very beginning, and evolves one's model as it becomes more structured.

## 7.1 Key aspects of our solution

We propose the concept of *malleable schema* to tackle the uncertainties in schema modeling. The key feature of malleable schemas is that a modeler can *capture* the important aspects of the domain at modeling time without having to commit to a very strict schema. A malleable schema begins the same way as a traditional schema, but at certain points gradually becomes vague. The vague parts of the schema can later evolve to have more structure, or can remain as such. Users can pose queries in which references to schema elements can be imprecise, and the query processor will consider closely related schema elements as well. We now briefly describe the technical issues.

First, we introduce *malleable elements* (classes and properties) into a malleable schema. The malleable elements look exactly the same as the other schema elements, except for the following (mostly conceptual) differences. While the name of a regular class or property is typically a carefully chosen string, the name of a malleable element can be made up of keywords or phrases, and is often obtained from external sources. Further, we support malleable element names to include also regular expressions (e.g., *Phone stands for workPhone, homePhone, daytimePhone, eveningPhone, etc.), or chains (e.g., address/zip states a zip is a component of an address).

Second, we introduce *imprecise references* to schema elements, denoted by $\sim K$, where $K$ is a class or property name. Consider a query that asks for the ∼name attributes of a Person instance, where the reference to name is imprecise. SEMEX will return the value of the name attribute, the value of the nickName attribute, the concatenation of the values of name/firstName and name/lastName, etc.

### 7.2 Related work

There has been a significant body of work on supporting keyword search in databases [48, 3, 49], result ranking [4, 45, 44], and approximate data values [64, 88, 32, 42, 20]. They all assume that the model of the data is precise, but want to add flexibility in the queries. In contrast, our goal is to allow the model itself to be imprecise in certain ways. Probabilistic databases allow imprecision about facts in the database, but the model of the domain is still a precise one.

The work closest to ours is the XXL Query Engine [87], where the queries allow for imprecise references to schema elements. The idea there is that the user will query a large *collection* of XML DTDs, and there is no unifying DTD for all of them. Malleable schemas, in contrast, offer a middle point between a collection of schemas/DTDs (or a corpus [61]) in a domain and a single crisp schema for that domain. The idea of a malleable schema is that someone *is* trying to create a schema for the domain, but in the process of doing so needs to introduce (possibly temporarily) some imprecision into the model. We expect to leverage some of the techniques in [87, 88] in our query processing engines.

## 8 Conclusions

Personal information management is increasingly attracting attention as an area of study. This proposal examines the problem from the data management perspective. In particular, we ask ourselves the big question: what is the right model for personal information and how can we leverage the model for data collection, data analysis, searching and browsing? We propose providing a logical view of one's personal data, consisting of meaningful objects and associations between the objects. We aim to build a prototype of a PIM system based on such a logical view.

A PIM system faces many challenges, including automatic data population, reference reconciliation, on-the-fly data integration, intelligent keyword search, and model personalization. Each has received significant attention in the literature, and is known to be rather challenging in itself. Our goal is to demonstrate how we can leverage the logical view of personal data to address these challenges, and eventually improve the user's productivity in her daily life.

## References


[1] S. Abiteboul, R. Agrawal, P. Bernstein, M. Carey, S. Ceri, B. Croft, D. DeWitt, M. Franklin, H. Garcia-Molina, D. Galwick, J. Gray, L. Haas, A. Halevy, J. Hellerstein, Y. Ioannidis, M. Kersten, M. Pazzani, M. Lesk, D. Maier, J. Naughton, H. Schek, T. Sellis, A. Silberschatz, M. Stonebraker, R. Snodgrass, J. Ullman, G. Weikum, J. Widom, and S. Zdonik. The lowell database research self assessment. *CoRR cs.DB/0310006*, 2003.

[2] D. Abrams, R. Baecker, and M. Chignell. Information archiving with bookmards: personal web space constrction and organization. In *CHI*, pages 41–48, 1998.

[3] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, 2002.

[4] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *cidr*, 2003.

[5] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating Fuzzy Duplicates in Data Warehouses. In *Proc. of VLDB*, 2002.

[6] D. Barreau and B. A. Nardi. 'finding and reminding' file organization from the desktop. *SIGCHI Bulletin*, 27(3), 1995.

[7] V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI*, pages 345–352, 2003.

[8] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering*, 36(3), 2001.

[9] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD*, 2004.

[10] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD*, 2003.

[11] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems Special Issue on Information Integration on the Web*, September 2003.

[12] R. Boardman and M. A. Sasse. 'stuff goes into the computer and doesn't come out' a cross-tool study of personal information management. In *CHI*, 2004.

[13] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7), 1998.

[14] V. Bush. As we may think. *The Atlantic Monthly*, July 1945.

[15] Y. Cai, X. Dong, A. Halevy, M. Liu, and J. Madhavan. Personal information management with Semex. In *SIGMOD*, 2005.

[16] D. Calvanese, S. Castano, F. Guerra, D. Lembo, M. Melchiorri, G. Terracina, D. Ursino, and M. Vincini. Towards a Comprehensive Framework for Semantic Integration of Highly Heterogeneous Data Sources. In *Proc. of the 8th Int. Workshop on Knowledge Representation meets Databases (KRDB2001)*, 2001.

[17] S. Castano, V. D. Antonellis, and S. D. C. di Vemercati. Global viewing of heterogeneous data sources. *IEEE Trans Data Knowledge Engineering*, 13(2), 2001.

[18] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proc. of SIGMOD*, 2003.

[19] W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration, 2002.

[20] W. W. Cohen. Data integration using similairty joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, 2000.

[21] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWEB*, pages 73–78, 2003.

[22] H.-H. Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proc. of VLDB*, 2002.

[23] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.

[24] A. Doan, Y. Lu, Y. Lee, and J. Han. Object matching for information integration: a profiler-based approach. In *IIWeb*, 2003.

[25] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Int. WWW Conf.*, 2002.

[26] X. Dong and A. Halevy. A Platform for Personal Information Management and Integration. In *CIDR*, 2005.

[27] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proc. of SIGMOD*, 2005.

[28] X. Dong, A. Halevy, E. Nemes, S. Sigurdsson, and P. Domingos. Semex: Toward on-the-fly personal information integration. In *IIWeb*, 2004.

[29] P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton. Extending document management systems with user-specific active properties. *ACM TOIS*, 18(2), 2000.

[30] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *SIGIR*, 2003.

[31] D. W. Embley. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *WIIW*, 2001.

[32] R. Fagin. Fuzzy queries in multimedia database systems. In *PODS*, 1998.

[33] I. P. Fellegi and A. B. Sunter. A theory for record linkage. In *Journal of the American Statistical Association,40*, pages 1183–1210, 1969.

[34] S. Fertig, E. Freeman, and D. Gelernter. 'finding and reminding' reconsidered. *SIGCHI Bulletin*, 28(1), 1996.

[35] M. Franklin, M. Cherniack, and S. Zdonik. Data management for pervasive computing: A tutorial. Tutorial at the 2001 VLDB Conference, 2001.

[36] E. Freeman and D. Gelernter. Lifestreams: a storage model for personal data. *SIGMOD Bulletin*, 1996.

[37] D. Gage. Lifelog.

[38] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative data cleaning: language, model, and algorithms. In *VLDB*, pages 371–380, 2001.

[39] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. Mylifebits: Fulfilling the memex vision. In *ACM Multimedia*, 2002.

[40] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity search in databases. In *Proc. of VLDB*, 1998.

[41] Google desktop search toolkit. http://desktop.google.com/, 2004.

[42] L. Gravano, P. G. Ipeirotis, H.V.Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, 2001.

[43] L. Gu, R. Baxter, D. Vickers, and C. Rainsford. Record linkage: current practice and future directions. http://www.act.cmis.csiro.au/rohanb/PAPERS/record_linkage.pdf.

[44] L. Guo, J. Shanmugasundaram, K. Beyer, and E. Shekita. Structured value ranking in update-intensive relational databases. In *ICDE*, 2005.

[45] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *SIGMOD*, 2003.

[46] B. He and K. C.-C. Chang. Statistical schema integration across the deep web. In *Proc. of SIGMOD*, 2003.

[47] M. A. Hernandez and S. J. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.

[48] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *VLDB*, 2003.

[49] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, 2002.

[50] L. Jin, C. Li, and S. Mehrotra. Efficient Record Linkage in Large Data Sets. In *DASFAA*, 2003.

[51] W. Jones, S. Dumais, and H. Bruce. Once found, what then?: a study of 'keeping' behaviors in personal use of web information. In *ASIST*, pages 391–402, 2002.

[52] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM Data Mining (SDM)*, 2005.

[53] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In *Proc. of SIGMOD*, 2003.

[54] V. Kaptelinin. Umea: translating interaction histories into project contexts. In *CHI*, pages 353–360, 2003.

[55] D. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A general-purpose information management tool for end users based on semistructured data. In *CIDR*, 2005.

[56] M. Kersten, G. Weikum, M. Franklin, D. Keim, A. Buchmann, and S. Chaudhuri. Panel: A database striptease, or how to manage your personal databases. In *Proc. of VLDB*, 2003.

[57] M. Lansdale. The psychology of personal information management. *Applied Ergonomics*, 19(1):458–465, 1988.

[58] M. L. Lee, T. W. Ling, and W. L. Low. Intelliclean: a knowledge-based intelligent data cleaner. In *SIGKDD*, pages 290–294, 2000.

[59] B. S. Lerner. A model for compound type changes encountered in schema evolution. *ACM TODS*, 25(1):83–127, 2000.

[60] W. Li and C. Clifton. Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural network. *Data Knowledge Engineering*, 33(1), 2000.

[61] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-basd schema matching. In *Proc. of ICDE*, 2005.

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

[63] T. W. Malone. How do people organize their desks? implications for the design of office information system. *ACM Transactions on Office Information Systems*, 4:42–63, 1986.

[64] A. Marian, S. Amer-Yahia, N. Koudas, and D. Srivastava. Adaptive query processing of top-k queries in XML. In *ICDE*, 2005.

[65] A. K. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.

[66] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm. In *Proc. of ICDE*, 2002.

[67] M. Michalowski, S. Thakkar, and C. A. Knoblock. Exploiting secondary sources for unsupervised record linkage. In *IIWeb*, 2004.

[68] R. J. Miller, L. M. Haas, and M. A. Hernandez. Schema mapping as query discovery. In *VLDB*, 2000.

[69] T. Milo and S. Zohar. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 1998.

[70] P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Pro. of Extending DataBase Technologies*, 2000.

[71] B. A. Nardi, S. Whittaker, E. Isaacs, M. Creech, J. Johnson, and J. Hainsworth. Integrating communication and information through contactmap. *Coummunications of the ACM*, 45(4):89–95, 2002.

[72] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. In *Science 130 (1959), no. 3381*, pages 954–959, 1959.

[73] L. Palopoli, D. Sacca, and D. Ursino. An automatic technique for detecting type conflicts in database schemas. In *CIKM*, pages 306–313, 1998.

[74] Parag and P. Domingos. Multi-relational record linkage. In *Proceedings of the Third Workshop on Multi-Relational Data Mining*, 2004.

[75] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS*, 2002.

[76] J. C. Pinheiro and D. X. Sun. Methods for linking and mining massive heterogeneous databases. In *SIGKDD*, 1998.

[77] J. E. Pitkow and C. M. Kehoe. Emerging trends in the www user population. *Communications of the ACM*, 39(6):106–108, 1996.

[78] D. Quan, K. Bakshi, D. Huynh, and D. R. Karger. User interfaces for supporting multiple categorization. In *INTERACT*, 2003.

[79] D. Quan, D. Huynh, and D. R. Karger. Haystack: A platform for authoring end user semantic web applications. In *ISWC*, 2003.

[80] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[81] G. Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[82] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning, 2002.

[83] F. Shipman, H. Hsieh, R. Airhart, P. Maloor, J. M. Moore, and D. Shah. Emergent structure in analytic workspaces: design and use of the visual knowledge builder. In *INTERACT*, pages 132–139, 2001.

[84] L. Tauscher and S. Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. *International Journal of Human-Computedr Studies*, 47:97–137, 1997.

[85] L. Tauscher and S. Greenberg. Revisitation patterns in world wide web navigation. In *CHI*, pages 399–406, 1997.

[86] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification, 2002.

[87] A. Theobald and G. Weikum. Adding relevance to XML. *Lecture Notes in Computer Science*, 2001.

[88] A. Theobald and G. Weikum. The index-based XXL search engine for querying XML data with relevance ranking. In *EDT*, 2002.

[89] S. Whittaker and C. Sidner. Email overload: exploring personal information management of email. In *CHI*, pages 276–283, 1996.

[90] W. E. Winkler. Using the em algorithm for weight computation in the fellegi-sunter model of record linkage. In *Section on Survey Research Methods*, pages 667–671. American Statistical Association, 1988.

[91] D. Wolber, I. Ranitovic, M. Kepe, and I. Sadreddin. Navigating the personal, shared, and historical webs. In *WWW*, 2003.

[92] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *Proc. of SIGMOD*, 2004.

[93] L. Xu and D. Embley. Discovering Direct and Indirect Matches for Schema Elements. In *DASFAA*, 2003.

[94] K. Zhang and D. Shasha. Approximate tree pattern matching. In *Pattern matching in strings, trees, and arrays*, pages 341–371, 1997.