iMeMex: A Platform for Personal Dataspace Management*

Marcos Antonio Vaz Salles Jens-Peter Dittrich

Institute of Information Systems
ETH Zurich
8092 Zurich, Switzerland
dbis.ethz.ch | iMeMex.org

ABSTRACT

Desktop computers provide thousands of different applications that query and store data in hundreds of thousands of files of different formats. Those files are stored in the local filesystem and also in a number of remote data sources, such as network shares or as attachments to emails. To handle this heterogeneous and distributed mix of personal information, data processing logic is re-invented inside each application. This results in an undesirable situation: most advanced data management functionality, such as complex queries, backup and recovery, versioning, provenance tracking, among others, is (at least partially) performed by end-users in tedious, manual tasks. To solve these problems we propose a software platform that brings *physical and logical data independence* to the desktop, freeing users from low-level data management activities. Unlike current relational DBMSs, this platform unifies data from several independent personal data sources without imposing semantic schema integration. It manages the complex dataspace [12] of one's personal information. We attack three major research challenges in the building of that platform: *(i)* definition of a data model that allows the integration of information in distinct representations and locations, *(ii)* design of a new search&query language over this data model along with algorithms for the efficient processing of complex queries, and *(iii)* formulation of an update model that enables soft durability guarantees, when compared to ACID properties, on data authored independently from the platform.

1. INTRODUCTION

In 1945, Bush [3] presented a vision of a personal information management system named *memex*. That vision has deeply influenced several advances in computing. Part of that vision led to the development of the *Personal Computer* in the 1980's. It also led to the development of *hypertext* and the *World Wide Web* in the 1990's. Since then, several projects have attempted to implement other memex-like functionality [13, 2, 4, 18]. Further, personal information management regained interest in the DB research community [15, 9, 8]. Moreover, it was identified as an important topic in the Lowell Report [1], discussed in a VLDB panel [19], considered in an NSF-sponsored workshop [17], debated in the SIGIR 2006 PIM workshop [22], and became topic of both SIGMOD 2005 keynotes [2, 21].

In spite of these previous efforts, we argue that a satisfactory solution has not yet been brought forward to the issues of *physical and logical data independence* on the desktop. Physical data independence relates to abstraction from the devices and formats in which data is represented. This is clearly not achieved by the simple data model of the current generation of file systems. Applications develop specific solutions that directly handle protocols to access the data (email, RSS/ATOM, network file system, etc) and also formats in which data is stored (XML, LaTeX, image and audio formats, etc). This creates application-specific data silos in which data management functionality, e.g., querying, updating, performing backup and recovery operations, are absent or re-invented. Logical data independence relates to the capability of defining views over the logical data model in which data is represented. It is also only partially achieved with current desktop systems, e.g. smart folders.

Personal Dataspaces. DBMS technology successfully resolved the physical and logical data independence problem for highly structured data, but not for the highly heterogeneous data mix present in personal information. Indeed, Franklin et al. [12] argue that today we rarely have a situation in which all the data that needs to be managed can fit nicely into a relational DBMS. Rather, most of the data will be authored independently from the DBMS and will not be in its full control. Franklin et al. introduce the term *dataspace* to describe this world of disparate, distributed and independently authored unstructured, semi-structured and structured data.

In this project we focus on *personal dataspaces*, that is the total of all personal information pertaining to a given individual. In contrast to the vision of [12], we propose one concrete *Personal Dataspace Management System (PDSMS)* implementation, named `iMeMex` (integrated memex). Unlike traditional information integration approaches, a PDSMS does not require *semantic data integration* before any data services are provided. Rather, a PDSMS is a *data co-existence approach* in which tighter integration is performed in *"pay-as-you-go"* fashion [12].

Current Status. The ultimate goal of the dissertation is to build the first publicly available PDSMS. The dissertation has so far one year of development and a research plan has been drawn for the next three years. In the first year of work, the Ph.D. student has helped to set the vision and context for the `iMeMex` project. As a result of this work, we have written a research proposal detailing the goals and work breakdown for the whole project. This proposal has been accepted by the Swiss National Science Foundation (SNF)[6] and supports two Ph.D. positions for a period of three years.

To evaluate our ideas, we have developed one first prototype of

*This work is partially supported by the Swiss National Science Foundation (SNF) under contract 200021-112115.

Proceedings of the VLDB2006 Ph.D. Workshop
Seoul, Rep of Korea, 2006

iMeMex. It was demonstrated in [8] and provided a traditional file system interface to explore arbitrary views over one's personal information. In parallel to the development of the prototype, we have defined our data model for representing personal information: the iMeMex Data Model (iDM). That model is presented in [7].

The current, second, version of the iMeMex platform extends the first prototype and incorporates the work on iDM. It offers a unified view on a set of personal data sources and allows basic query processing on that view. Our current implementation (Java 1.4) contains about 215 classes and 22,000 lines of code.

Outline. We present related personal information management approaches in Section 2. We proceed by discussing the research challenges involved in building a PDSMS in Section 3. We then describe in Section 4 some of our solutions to these challenges, which consist of: (1) a unified data model for personal information (Section 4.1); (2) a flexible query language that operates on this model, along with techniques for efficient query processing (Section 4.2); (3) an update model for a PDSMS which includes mechanisms for recovery and versioning of all data present in a personal dataspace (Section 4.3); and (4) the architecture of a PDSMS, which integrates all of the previous contributions into a unified framework (Section 4.4). Finally, we conclude in Section 5.

2. STATE OF THE ART

As we approach an age in which each computer user will face the challenge of managing her own personal terabyte, PIM research has obtained renewed interest in a variety of areas, such as HCI, IR and data management [17]. Due to space limitations, we only comment on a few solutions in this section. Current operating systems have been amended in the past years to include *full-text search appliances*, such as Google Desktop, Apple Spotlight, and Phlat [4]. These systems offer an intuitive keyword search interface, sometimes augmented by generic metadata (modification date, author, etc). Their data models, however, are unable to represent structural information inside documents. A PDSMS, in contrast, enriches keyword and property search with advanced structural querying.

Systems such as SEMEX [9] and Haystack [18] allow users to *browse by association*. They employ an ETL cycle to extract information from desktop data sources into a repository and represent that information in a domain model (ontology). The domain model is a high-level mediated schema over the personal information sources. These systems focus on creating a queryable, however non-updatable, view on the user's personal information. In contrast, a PDSMS offers support for not only advanced querying and browsing but also for updating information in the underlying personal dataspace whenever possible. In fact, all of the systems above may be thought of as applications on top of a PDSMS.

Other systems offer tools to ease the *management of personal data*. Lifestreams [13] organizes all personal documents in a timeline. In Placeless Documents [11], users may tag their documents with active properties, such as "backup" or "replicate", and the appropriate actions will be carried out by the system. MyLifeBits [2] models each piece of information as resources and allows these resources to be annotated and organized in collections. Microsoft WinFS [23], now discontinued[1], represented information in an item data model which is a subset of the object-oriented data model and offered a basic class library to represent data items commonly found in user desktops. Like MyLifeBits, WinFS based storage of items on a relational DBMS. All of these systems need full control of the data to offer features such as backup&recovery. In contrast, a PDSMS enables data to be authored and updated independently by the interfaces offered by the underlying data sources. Further, in these systems, advanced PDSMS queries that bridge structural information across the inside-outside file boundary are not available (see Example 1).

[1]The downloadable beta as well as all other preliminary information about WinFS were recently removed from its web-site [23].

3. RESEARCH CHALLENGES

In the following, we discuss major research challenges that are targeted by the Ph.D. on the iMeMex PDSMS.

Challenge 1 (Representing Personal Information). A major research challenge of managing personal information is dealing with its heterogeneity. Heterogeneity relates to data models and formats used to represent the information. It also relates to the data sources in which that information is available and to the mechanisms available for data delivery (push/pull). Let's consider an example:

EXAMPLE 1 (INSIDE AND OUTSIDE FILES) Users organize their workspaces in folder hierarchies and use applications to store information inside files. Each file is an independent data cage in which complex structural representations may be stored. Consider the following query: "Show me all LaTeX 'Introduction' sections pertaining to project PIM that contain the phrase 'Personal Information"'. With current technology, this query cannot be issued in one single request by the user as it has to bridge the *inside-outside file boundary*. The user may only search the file system using simple system tools like grep, find, or a keyword search engine. However, these tools may return a large number of results which would have to be examined manually to determine the final result. Even when a matching file is encountered, then, for structured file formats like Microsoft PowerPoint, the user typically has to conduct a second search inside the file to find the desired information [4]. Moreover, state-of-the-art operating systems do not support at all exploitation of structured information inside the user's documents. □

Ideally, we would like to have a common representation for all personal information in different data models and sources. This common representation (or view) would enable queries that ignore how the data is stored or where it is located. In addition, we should be able to construct that view *without* performing labor-intensive semantic schema integration. Rather, we would like to perform lightweight *data model integration* and leave expensive semantic integration to be carried out in a "pay-as-you-go" fashion.

Challenge 2 (Querying Personal Information). Once we have an integrated view on one's personal dataspace, the next natural challenge is how to query this view. Users have traditionally employed browsing (i.e., neighborhood expansion) and keyword queries to explore their data. Ideally, we would like to provide one single search&query language to *analyze* and *modify* all data in a personal dataspace. This query language should allow impreciseness in query formulation and also integrate ranking of query results. Further, advanced functionality, such as branching expressions and joins, should be available. Note that expressions written in this language should be processed with interactive response times.

Challenge 3 (Updating Personal Information). Given a unified view of all personal information, another important challenge is to provide means to update that personal information through that unified view. Current desktop search engines (DSEs) are read optimized systems. DSEs are able to *detect* updates made to the data sources and to *incorporate* those updates into their index structures. They have, however, two important drawbacks: (1) DSEs do not allow applications to perform updates on the data sources *through* the DSEs' interfaces, and (2) DSEs do not offer any update guarantees on the underlying data, such as *durability* (e.g., to allow recovery of past images of the data).

DBMSs, on the other hand, provide update interfaces and also strict transactional ACID guarantees, but demand a high price for them: full control of the data. In contrast to both approaches, a PDSMS occupies the middle-ground between a read-only DSE (without update guarantees) and a write-optimized DBMS (with strict ACID guarantees). Guarantees may vary according to the interfaces offered by the data sources managed by the PDSMS. To enforce soft durability guarantees, a PDSMS must provide algorithms for backup&recovery, versioning, update handling, among others.

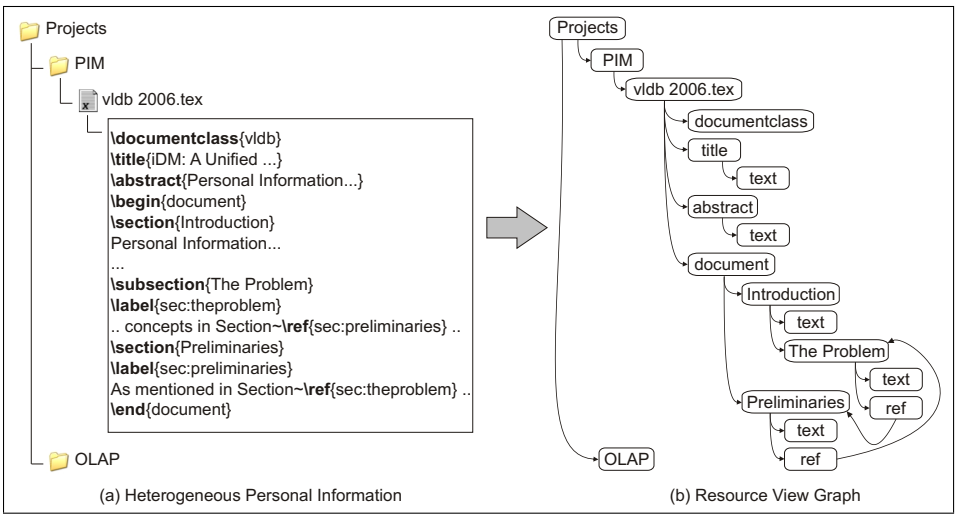

Figure 1: iDM represents heterogeneous information in a single resource view graph

4. OUR APPROACH

In this section, we provide more details on our previous and ongoing work on the iMeMex PDSMS. First, we discuss our solutions for each of the challenges described in the previous section. We then conclude by presenting the iMeMex PDSMS architecture, which serves as a framework to deploy the presented techniques.

4.1 Representing Personal Information

Figure 1(a) depicts the situation described in Example 1. It shows a files&folders hierarchy with information on research projects of one of the authors. Note that arbitrary cyclic graph structures may naturally arise in data inside files. In the LaTeX document "vldb 2006.tex", for example, inside the subsection "The Problem", there is a reference to the section "Preliminaries" and vice versa. An extended example of such occurences is provided in [7].

Why XML is not Enough. Ideally, files&folders as well as the structure inside files should be represented into the same logical data model. One could try to employ XML technology to address this challenge of representation heterogeneity. In fact, we followed that approach in [8]. Unfortunately, XML is associated to *both* a logical data model *and* a physical markup to represent this logical model. This means that the manipulation of XML views is coupled with serialization concerns. Recent work has identified this gap, e.g. [20, 16, 14], and argues in favor of clearly separated *logical data models* supporting more advanced features, e.g. multiple hierarchies [16]. However, none of the existing approaches is sufficient to naturally represent the complex, possibly infinite, distributed and lazily computed information graph encountered in a personal dataspace. Therefore, we have decided to represent all personal information based on a novel, more powerful, *logical* data model: the *iMeMex Data Model (iDM)*.

Resource View Graph. We briefly sketch a few characteristics of iDM in this section; full details are provided elsewhere [7]. iDM enables a logical representation of a personal dataspace, as shown in Figure 1(b). The main features of iDM are:

- **Resource Views:** in iDM, all personal information is represented by fine-grained *resource views*. A resource view is made of components that express *structured*, *semi-structured* and *unstructured* pieces of the underlying data. For instance, resource views may represent nodes in a files&folders hierarchy as well as elements in an XML, LaTeX or other office document. Other than that, we use resource views to uniformly represent email messages, email attachments, infinite data streams, relational data, RSS/ATOM messages, bookmarks, query results, calls to web services and many others [7]. The granularity at which resource views are represented is determined by a set of *plugin components* in our system architecture (see Section 4.4).
- **Graphs:** resource views in iDM are linked to each other forming directed *graph structures*. In Figure 1(b), we show the resource view graph that corresponds to the personal data in Figure 1(a). In that graph, there is no inside-outside file boundary. All structural elements (folders, sections, subsections, etc) are represented in the same model and queries may address them uniformly. Note that cycles may naturally arise in that graph (in this example as a consequence of section cross referencing).
- **Intensional Data:** any given resource view or parts of a resource view graph may be either materialized (i.e., extensional data) or computed on demand as the result to a query or to a remote web service invocation (i.e., intensional data [20]). This is in sharp contrast to static data models such as XML.
- **Stream Support:** another important feature of our model is the ability of resource views to contain *finite* as well as *infinite* components. Infinite resource view components are used to represent data streams (e.g., RSS, publish/subscribe) and content streams (e.g., audio and video) in our model.

In our approach, the notion of impreciseness is included in our query language, briefly discussed in Section 4.2.

Data Model Instantiations. A resource view is given by the following four formal components:

name η	Name of the resource view.
tuple τ	List of attribute value pairs $((name_0, value_0), (name_1, value_1), \ldots)$.
content χ	(in)finite In-/Output of content (e.g. text).
group γ	References to other resource views. - S: (in)finite set $\{\ldots\}$ - Q: (in)finite ordered sequence $\langle\ldots\rangle$

We use *resource view classes* to constrain resource view components. Resource view classes allow integration of data from diverse data models into iDM without requiring time consuming semantic schema integration. A resource view V_i of class C is denoted by V_i^{C}. Similarly, its components are denoted by η_i^{C}, τ_i^{C}, χ_i^{C}, and γ_i^{C}.

We show in Table 1 how our model may be constrained to represent files, folders and the core subset of XML. We denote the name of an underlying data item i by N_i, attribute-value pairs associated to it by a schema W and a tuple T_i, and its content by C_i.

Resource View Class		Resource View Components Definition				
Description	Name	η_i^C	τ_i^C	χ_i^C	γ_i^C	
File	file	N_f	$\langle W_{FS}, T_f \rangle$	C_f	S	$\varnothing$
					Q	$\langle\rangle$
Folder	folder	N_F	$\langle W_{FS}, T_F \rangle$	$\langle\rangle$	S	$\{V_1^{\text{child}}, \ldots, V_m^{\text{child}}\}$ $\text{child} \in \{\text{file}, \text{folder}\}$
					Q	$\langle\rangle$
XML text node	xmltext	$\langle\rangle$	$\langle\rangle$	C_t	S	$\varnothing$
					Q	$\langle\rangle$
XML element	xmlelem	N_E	$\langle W_E, T_E \rangle$	$\langle\rangle$	S	$\varnothing$
					Q	$\langle V_1^{\text{child}}, \ldots, V_n^{\text{child}} \rangle$ $\text{child} \in \{\text{xmltext}, \text{xmlelem}\}$
XML document	xmldoc	$\langle\rangle$	$\langle\rangle$	$\langle\rangle$	S	$\varnothing$
					Q	$\langle V_{\text{root}}^{\text{xmlelem}} \rangle$
XML File	xmlfile	N_f	$\langle W_{FS}, T_f \rangle$	C_f	S	$\varnothing$
					Q	$\langle V_{\text{doc}}^{\text{xmldoc}} \rangle$

Table 1: Resource View Classes for files&folders and XML

The instantiations shown in Table 1 allow the creation of resource view graphs as the one shown in Figure 1(b). Our data model is, however, much more powerful: instantiations for relations and data streams, as well as a more rigorous discussion on intensional aspects [20] of iDM are presented in [7].

4.2 Querying Personal Information

The iMeMex PDSMS should offer querying services on the resource view graph representing all of one's personal dataspace. In the following, we discuss our ongoing work and open issues on query specification and processing.

Personal Dataspace Query Language. We propose a new search& query language for schema-agnostic querying of a resource view graph: the iMeMex Query Language (iQL). The definition of the iQL syntax and associated semantics is work in progress. In our current implementation, the syntax of iQL is a mix between typical search engine keyword expressions and XPath navigational restrictions. The semantics of our language are, however, much different than those of XPath and XQuery. Our language's goal is to enable querying of a resource view graph that has not necessarily been submitted to expensive schema integration. Therefore, as in search technology, we account for impreciseness in query formulation. For example, by default, when an attribute name is specified (e.g. size > 10K), we do not require exact matches on the (implicit or explicit) schema for that attribute, but rather return fuzzy, ranked results for the resource views that better match the specified conditions (e.g. size, fileSize, docSize). This allows us to define malleable schemas as in [10]. Other important features of iQL are the ability to reflect structural constraints, e.g. to explore the context or neighborhood of items, the definition of extensible algebraic operations like joins and grouping, and the specification of updates to the resource view graph.

Indexing Techniques. In our current implementation of iMeMex, we index all components of every resource view created in the system. This full indexing strategy follows the intuition that the PIM environment shares with data warehousing the characteristic of low update rates, allowing us to trade space and indexing time for query performance. The information from each component of a resource view (e.g., name or group of related resource views) goes to a different index and we perform intersects to process conditions on several components. We plan to investigate whether it pays off to have integrated index structures for various resource view components. In contrast to traditional XML indexing, our index structures must operate in a general graph data model on possibly infinite data.

Cost-based Optimization. Cost-based optimization (CBO) is one key technique to provide interactive response times in read-mostly environments. We are planning to build a CBO for iMeMex to account for trade-offs in the usage of alternative query plans, e.g., to consider join orders and different access methods.

Neighborhood Queries. Providing *context* is key to enable exploration of query results [5]. Thus, it is a common pattern to query the neighborhood of objects returned from a previous query. One alternative to speed-up such queries is to keep their results materialized in a special index structure. This index may cover only the immediate neighborhood of each resource view or it may be extended to include other reachable resource views. We plan to evaluate how much context should be kept in the index structure to account for trade-offs in querying speed, indexing time and update processing.

4.3 Updating Personal Information

The iMeMex PDSMS should offer soft durability guarantees on updates made through its interface or via the APIs of the underlying data sources bypassing iMeMex. In the following, we discuss our ideas for tackling that challenge.

Dataspace Update Model. We plan to design an update model for the iMeMex PDSMS that accounts for the fact that data may be independently updated via the APIs of the underlying data sources. In this scenario, ACID guarantees are too strict, once the iMeMex PDSMS may be notified of updates "after the fact". Nevertheless, we believe that classical database logging techniques may be adapted to this setting to provide softer recovery guarantees (e.g., all items updated more than 5 min ago may be recovered).

Versioning. In relational systems, previous versions of a given tuple may be reconstructed from the database log (see e.g. "time travel" feature of Oracle). However, personal items are typically more heavyweight than relational tuples, as they may have medium to large content. An alternative to logging would be to keep an independent versioning subsystem (e.g. Subversion) to track content evolution. We plan to investigate how to integrate versioning into our update model for personal information and also whether there are profitable interactions with the techniques chosen for recovery.

Write back. Updates to personal information may be performed via the API of a given data source or via iMeMex's API. In the latter case, one must write the data back to the affected data sources. If the data is not already present in any data source, iMeMex must decide in which subsystem(s) it is most suitable to be represented.

Distribution. When a user has several devices, it is natural to ask how to manage several iMeMex instances and coordinate distributed query and update processing among these instances. We believe that the many challenges of this scenario exceed the scope of the current Ph.D. work. Those challenges will be tackled by a separate Ph.D. thesis as part of the iMeMex project.

4.4 iMeMex PDSMS Architecture

We present in this section the current architecture of the iMeMex PDSMS, which serves as a framework for all of the previously discussed technical contributions. We also indicate points of ongoing work in which the architecture will be extended.

The core idea of iMeMex is to implement a logical layer that abstracts from the underlying subsystems and data sources, such as file systems, email servers, network shares, music streams, RSS feeds, etc. That logical layer does not take full control of the data, so it may be bypassed by applications. Figure 2 depicts that layer and its current implementation in iMeMex.

iMeMex contains two important sublayers: iQL Query Processor and Resource View Manager. The main task of the *iQL Query Processsor* is to translate incoming iQL queries and to create query plans for those queries. Our current implementation is based on rule-based query optimization. We plan to invest in cost-based optimization techniques as part of future work.

The Resource View Manager (RVM) is the central instance to

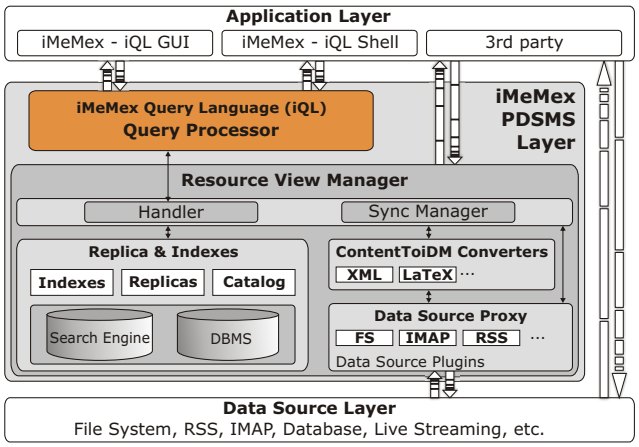

Figure 2: `iMeMex` PDSMS architecture

managing resource views. Its major components are: Data Source Proxy, ContentToiDMConverters, Replica&Indexes Module, and Synchronization Manager. We describe them in the following.

Data Source Proxy. Provides connectivity to the distinct types of subsystems. It contains a set of *Data Source Plugins* that represent the data from the different subsystems (e.g., file systems, RSS, IMAP, databases, etc) as an initial iDM graph.

ContentToiDMConverters Module. Enriches the iDM graph provided by the data source proxy. This is achieved by converting resource view content to iDM subgraphs that then reflect structural information (e.g., in LaTeX, XML, etc). The result is an iDM graph such as the one presented in Section 4.1.

Replica&Indexes Module. Materializes mappings between resource view identifiers and resource view components (e.g., name or group of related resource views) to accelerate query processing. A mapping from resource view identifiers to copies of component instances is termed a **replica**. The inverse mapping is termed an **index**. Currently, our implementations of replicas and indexes are based on a DBMS (Apache Derby) for structured information, such as attribute-value pairs and resource view connections, and on inverted keyword lists (Apache Lucene) for textual information, such as names and text content. We plan to extend this module to provide specialized index structures as discussed in Section 4.2.

Synchronization Manager. Monitors registered data sources for changes. When a data source is registered at the RVM, the Synchronization Manager analyzes the data found on the data source and sends each resource view definition to the Replica&Indexes Module. The Synchronization Manager also subscribes to update notifications from the data source. As a consequence, updates performed on the data source bypassing the RVM layer are then immediately considered by the Synchronization Manager and the Replica&Indexes Module. If the data source does not offer update notifications, the Synchronization Manager generates them based on a generic polling facility. We will extend this module to incorporate recovery and versioning techniques, as described in Section 4.3.

5. CONCLUSION

Personal Information Management has become a key necessity for almost everybody. Reflecting this prominence, considerable attention has been given to PIM research in the recent past. At the same time, it has become clear that what is missing is a unified approach to bring physical and logical data independence to the management of one's personal dataspace. We address three major research challenges in the pursuit of this goal. First, we define a new data model capable of representing the heterogeneous data mix found in personal dataspaces. As one application of our model we bridge the artificial boundary that separates inside and outside files. Second, we are working on a new search&query language that operates on our data model. The processing of expressions in this language calls for the design of efficient techniques, e.g. for indexing and neighborhood querying. Third, we are working on a dataspace update model. That model will provide soft durability guarantees, write-back to data sources as well as detection of changes made on data sources bypassing `iMeMex`. We plan to design integrated recovery and versioning techniques to support our update model. By building the first publicly available PDSMS, we believe that we make a significant contribution to the development of advanced PIM applications.

6. REFERENCES

[1] S. Abiteboul, R. Agrawal, P. A. Bernstein, and others. The Lowell Database Research Self Assessment. *The Computing Research Repository (CoRR)*, cs.DB/0310006, 2003.

[2] G. Bell. MyLifeBits: a Memex-Inspired Personal Store; Another TP Database (Keynote). In *ACM SIGMOD*, 2005.

[3] V. Bush. As we may think. *Atlantic Monthly*, 1945.

[4] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin. Fast, flexible filtering with Phlat — Personal search and organization made easy. In *CHI*, 2006.

[5] J.-P. Dittrich, P. M. Fischer, and D. Kossmann. AGILE: Adaptive Indexing for Context-Aware Information Filters. In *ACM SIGMOD*, 2005.

[6] J.-P. Dittrich and D. Kossmann. iMeMex: A Unified Approach to Personal Information Management. In *SNF project under contract 200021-112115*.

[7] J.-P. Dittrich and M. A. V. Salles. iDM: A Unified and Versatile Data Model for Personal Dataspace Management. In *VLDB*, 2006.

[8] J.-P. Dittrich, M. A. V. Salles, D. Kossmann, and L. Blunschi. iMeMex: Escapes from the Personal Information Jungle (Demo Paper). In *VLDB*, 2005.

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

[10] X. Dong and A. Y. Halevy. Malleable Schemas: A Preliminary Report. In *WebDB*, pages 139–144, 2005.

[11] P. Dourish et al. Extending Document Management Systems with User-Specific Active Properties. *ACM Transactions on Information Systems (TOIS)*, 18(2):140–170, 2000.

[12] M. Franklin, A. Halevy, and D. Maier. From Databases to Dataspaces: A New Abstraction for Information Management. *SIGMOD Record*, 34(4):27–33, 2005.

[13] E. Freeman and D. Gelernter. Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, 25(1):80–86, 1996.

[14] J. Graupmann, R. Schenkel, and G. Weikum. The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents. In *VLDB*, 2005.

[15] A. Halevy et al. Crossing the Structure Chasm. In *CIDR*, 2003.

[16] H. V. Jagadish, L. V. S. Lakshmanan, M. Scannapieco, D. Srivastava, and N. Wiwatwattana. Colorful XML: One Hierarchy Isn't Enough. In *ACM SIGMOD*, 2004.

[17] W. Jones and H. Bruce. A Report on the NSF-Sponsored Workshop on Personal Information Management, Seattle, Washington, 2005.

[18] D. R. Karger et al. Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. In *CIDR*, 2005.

[19] 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 *VLDB*, 2003.

[20] T. Milo, S. Abiteboul, et al. Exchanging Intensional XML Data. In *ACM SIGMOD*, 2003.

[21] T. Mitchell. Computer Workstations as Intelligent Agents (Keynote). In *ACM SIGMOD*, 2005.

[22] SIGIR PIM 2006. `http://pim.ischool.washington.edu/pim06home.htm`.

[23] `http://msdn.microsoft.com/data/WinFS/` WinFS.