`iMeMex`: A Platform for Personal Dataspace Management*

Position Paper

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 unfortunate 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 named `iMeMex` that brings *physical and logical data independence* to the desktop, freeing users from low-level data management considerations. Unlike in relational DBMSes, our platform does not assume full control of the data, but rather manages the complex dataspace [11] of one's personal information. We discuss several research challenges encountered building such a platform: (i) the definition of a unified data model that allows the integration of information in distinct representations and locations without requiring semantic data integration, (ii) the development of a new search&query language over this data model along with algorithms for the efficient processing of complex queries and (iii) the need for soft update and recoverability techniques to provide durability and consistency of personal information.

1. INTRODUCTION

In 1945, Bush [3] presented a vision of a personal information management system named *memex*. The memex would allow an individual to browse his information through associations among concepts. It would also allow sharing of information with other individuals. That vision has deeply influenced several progresses in computing. Part of that vision led to the development of the *Personal Computer* in the 1980ies. It also led to the development of *hypertext* and the *World Wide Web* in the 1990ies. Since then, several projects have attempted to implement other memex-like functionalities [12, 2, 4, 17]. In addition, personal information management regained interest in the database research community [14, 9, 8]. Moreover, it was identified as an important topic in the Lowell Report [1], discussed in a VLDB panel [18], and became topic of both SIGMOD 2005 keynotes [2, 21].

In spite of all these previous efforts, we argue that a satisfactory solution has not yet been brought forward to the issues of *physical and logical data independence* in 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. Unfortunately, applications tend to develop specific solutions to directly handle protocols to access the data (email, RSS/ATOM, network share, 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 application or user-centric views over the data model that is used to represent data. It is also only partially implemented with current desktop technology.

Personal Dataspaces. Although DBMS technology successfully resolved the physical and logical data independence problem for highly structured data, it is no coincidence that the problem remains unsolved for the highly heterogeneous data mix present in personal information. Franklin et al. [11] argue that today we rarely have a situation in which all the data that needs to be managed can fit nicely into a conventional relational database management system (DBMS). Rather, most of the data will be authored independently from a DBMS and will not be in its full control. Franklin et al. term this world of disparate, distributed and independently authored unstructured, semi-structured and structured data a *dataspace*.

In this project we focus on *personal dataspaces*, that is the total of all personal information pertaining to a certain person. In contrast to the vision presented in [11], we propose a 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 a *pay-as-you-go* fashion [11].

Current Project Status. The ultimate goal of the `iMeMex` project is to build the first publicly available PDSMS. The `iMeMex` project has so far 15 months of development. In the first year of work, the vision for the `iMeMex` project was developed. To evaluate our ideas, we have implemented a prototype of the `iMeMex` platform. This prototype was key in gathering requirements and understanding the challenges in constructing a PDSMS. It was demonstrated in [8]. It provided a traditional file system interface to explore arbitrary file system views over one's personal information. In addition, we have written a research proposal and project plan detailing the goals and work breakdown for the entire project. This proposal has been ac-

cepted by the Swiss National Science Foundation (SNF) supporting two Ph.D. positions for a period of three years [6].

In parallel with the implementation of the first system prototype, we have developed a unified data model for representing personal information, the `iMeMex` Data Model (iDM). Our data model iDM is described in an upcoming research paper [7].

The current, second, version of the `iMeMex` platform capitalizes on the experience gained with the development of the prototype and incorporates the research work done on iDM. Our implementation of the `iMeMex` PDSMS currently contains about 215 classes and 22,000 lines of code (excluding documentation). It is based on Java 1.4. The project [23] includes one senior research associate, two full time Ph.D. students, one student assistant and twelve M.Sc. students (seven of which have now completed their subprojects).

2. RELATED WORK

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 [16]. Due to space limitations, we only comment on a few solutions in this section. Modern 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 navigation (e.g., modification date, author, etc). Their data models, however, are unable to explore structural information inside documents. A PDSMS, in contrast, enables advanced querying, enriching keyword and property search with advanced structural querying.

Systems such as SEMEX [9] and Haystack [17] 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 must offer support for not only advanced querying and browsing, but also for updating information in the underlying personal dataspace whenever possible.

Other systems offer tools to ease the *management of personal data*. Lifestreams [12] bases the visualization of information on time. In Placeless Documents [10], 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 permits resources to be annotated and organized in collections. Microsoft WinFS — which was recently stopped [22] by Microsoft and will not be continued as a separate project — represented information in an item data model which is a subset of the object-oriented data model and offers a basic class library to represent data items commonly found in user desktops. Both MyLifeBits and WinFS base storage of items on a relational DBMS. In order to offer features such as backup&recovery, all of these systems need full control of the data. 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 (see Example 1) are not available.

3. RESEARCH CHALLENGES

In the following, we discuss three key challenges that are addressed by our work on the `iMeMex` Personal Dataspace Management System.

Challenge 1 (Representing Personal Information). A major challenge of managing personal information is dealing with its heterogeneity. Heterogeneity relates to data models and formats used to represent personal information. It also relates to the data sources in which that information is available and to the mechanisms available for data delivery (push/pull). Lets consider a simple 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 that *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 — including the upcoming Vista — do not support at all exploitation of structured information inside the user's documents. □

Ideally, we would like to have a single, unified representation for all personal information that bridges the divide between different data models and data representations. This unified representation (or view) would enable queries that ignore *which system* is used to store the personal information, *which format* is used and *where* the data is located. That unified view should not require any semantic integration efforts as required by traditional information integration approaches. We present our solution to this challenge in Section 4.

Challenge 2 (Querying Personal Information). Once we have an integrated view on one's personal dataspace, the next natural challenge is how to query that view. In order to efficiently query iDM, several research challenges have to be considered:

Language Specification. Traditionally, users have employed browsing (i.e., neighborhood expansion), path expressions (e.g., using cmd or tcsh) and keyword queries to explore their data. Ideally, we would like to provide one single search&query language to analyze and *modify* data available in a personal dataspace. This unified search&query language should integrate extensible ranking schemes and also allow impreciseness in query formulation. In addition, we would also like to include similarity operators, e.g., to allow fuzzy specification of attribute names to define malleable schemas [9]. Further, our language should also be able to reflect structural constraints, e.g. to explore the context or neighborhood of items. Finally, we are planning to integrate extensible algebraic operations like joins and grouping as well as update capabilities.

Indexing Techniques. A PDSMS should process queries with interactive response times. Therefore efficient dataspace indexing techniques have to be developed. The indexes should allow efficient retrieval of data based on fuzzy, similarity as well as structural predicates. Further, the index structures must operate on large graph data in distributed environments.

Cost-based Optimization. Cost-based optimization (CBO) is one key technique to providing 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 or different access methods. In addition, for distributed instances of `iMeMex` the trade-off between query and data shipping [19] has to be considered by the CBO.

Context Queries. Providing *context* is key to enable browsing and further exploration of query results [5]. This means that it is a common pattern to query the neighborhood (or context) of objects returned from a previous query. Processing these kind of queries is a challenging task. One alternative to speed-up such queries is to keep their results materialized in a special index structure. However, other techniques exist and we are currently evaluating their trade-offs against materialization.

Challenge 3 (Updating Personal Information). Given a unified view of all personal information, the next challenge is to provide means to update that personal information using that unified view. The current generation of desktop search engines (DSE) does not offer any means to update the underlying data. DSEs are restricted to read-only querying. Data may only be changed by *directly* accessing the underlying data sources. For this reason, a DSE cannot offer any update guarantees, such as *durability* or *consistency*. DBMSes, on the other hand, provide 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 any update capabilities) 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. The development of PDSMS update mechanisms poses several challenges:

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 bypassing `iMeMex`. In this scenario, ACID guarantees are too strict since the `iMeMex` PDSMS may be notified of updates "after the fact". Nevertheless, we believe that classical database recovery techniques may be adapted to this setting to provide softer backup and recovery guarantees (e.g., all items updated more than 5 min ago may be recovered). The recovery mechanisms also have to work for dataspaces backed up by distributed instances of `iMeMex`.

Write back. Updates to personal information may be performed via the API of a given data source or directly via `iMeMex`'s API. Therefore, we must architect our PDSMS supporting and recognizing updates in the underlying data sources. Moreover, if updates are performed via `iMeMex`'s API, `iMeMex` has to write the data back to the affected data sources. In that case, the PDSMS should decide in which subsystem(s) it is most suitable to be represented.

Versioning. In a relational DBSMS, 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 account for content versioning. We plan to investigate how to integrate versioning into our update model and also whether there are profitable interactions with the techniques chosen for recovery (e.g. logging).

Distribution. A user may have several devices, e.g. laptop, PC, and handheld. The `iMeMex` PDSMS has to support dataspaces that are backed up by several distributed instances. Different instances are connected to form a single dataspace by using a name server. Data exchange is then performed in a peer-to-peer fashion. Note that distributed dataspaces also pose challenges related to security and privacy. We believe that the latter aspects are key to convincing end-users to trust the services offered by a PDSMS implementation.
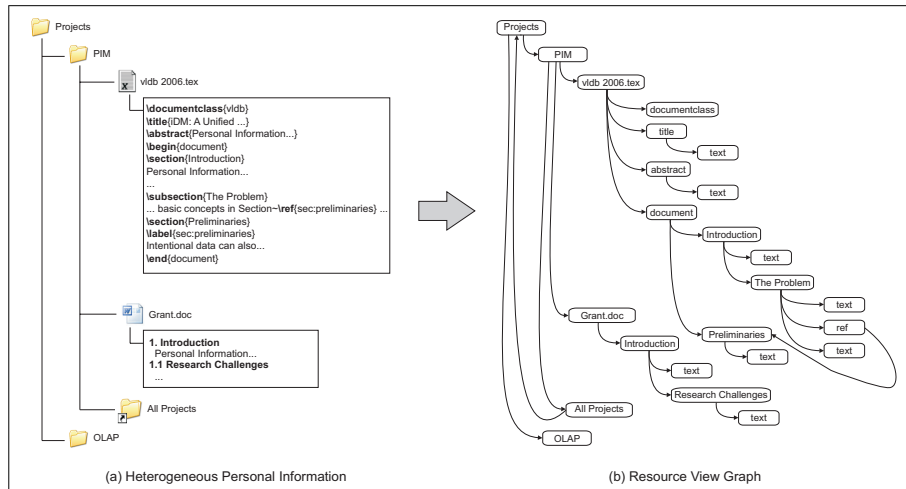

Figure 1: iDM represents heterogeneous information in a single resource view graph. That resource view graph represents the whole personal dataspace.

4. iDM: `iMeMex` DATA MODEL

This section sketches our solution to Challenge 1, i.e., how to find a unified representation of personal information. Our solutions to the other challenges cannot be presented here due to space constraints. We refer the reader to [8, 7].

Figure 1(a) depicts the situation described in Example 1. It shows a files&folders hierarchy with information on some research projects of the author. Note that the folders are organized in a graph structure, as the folder "All Projects" is a link to the top-level "Projects" folder. The data inside files is also organized in a graph structure. Further, in the LATEX document "vldb 2006.tex", inside the subsection "The Problem", there is a reference to section "Preliminaries".

4.1 Why XML is not Enough

Ideally, files&folders as well as the structure inside files should be represented by 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, 15, 13], and argues in favor of clearly separated *logical data models* supporting more advanced features, e.g. multiple hierarchies [15]. 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 decided to represent all personal information based on a more powerful, *logical* data model: the *iMeMex Data Model (iDM)*.

4.2 Resource View Graph

We briefly sketch a few characteristics of iDM in this section; full details are provided elsewhere [7].

Resource View. In iDM, all personal information is represented by *resource views*. A resource view consists of components that express *structured*, *semi-structured* and *unstructured* pieces of the underlying data. For instance, every file or folder in a files&folders hierarchy as well as every element in an XML, LATEX or other office document is represented in iDM by one distinct resource view. Other than that we use resource views to uniformly represent email messages, email attachements, relational data, infinite data streams, RSS/ATOM messages, bookmarks, query results, calls to web services and many others [7].

Graphs. Resource views in iDM are linked to each other forming directed *graph structures*. Recall, that Figure 1(a) shows some personal data found in a user's dataspace. That data is represented as a resource view graph as shown in Figure 1(b). In that resource view graph, there is no inside-outside file boundary anymore. All structural elements (folders, sections, etc.) are represented in the same model. Therefore queries may address them uniformly.

Intensional Data. It is important to stress that any given resource view or parts of a resource view graph may be either materialized (extensional data) or computed on demand by computing the result to a query or calling a remote web service (intensional data [20]). This is in sharp contrast to static data models such as XML.

Stream Support. Another important feature of our model is that resource views may 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.

Resource View Classes. We have defined resource view classes to constrain iDM to represent data available in traditional data models, such as files&folders, the relational model, XML, streamed sources, etc. Resource view classes are a mechanism that allow on-the-fly data integration from diverse data models into iDM without requiring time consuming semantic schema integration [7, 11].

Implementation. The resource view abstration is central to the implementation of the `iMeMex` PDSMS 2.0. That mechanism is also key to providing a unified query mechanism in a personal dataspace. In our current implementation of `iMeMex`, we apply a full indexing strategy. It follows the intuition that the PIM environment shares with data warehousing the characteristic of low update rates. This allows us to trade space and indexing time for query performance. The different components of resource views are indexed by separate indexes and we perform intersect operations to process conditions on several components. We plan to investigate whether it pays off to provide integrated index structures for various resource view components. Details on the architecture, indexing and query processing strategies are not presented here due to space constraints.

5. CONCLUSION

Personal Information Management has become a key necessity of almost everybody. 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 create physical and logical data independence to enable a *personal dataspace*. We address three major research challenges in the pursuit of this goal. First, we define a unified data model capable of representing the heterogeneous mix of information 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

query language that operates on our data model. The processing of expressions in this language calls for the design of new techniques, e.g. for indexes and neighborhood queries. Third, we are working on a dataspace update model. That model will include soft durability guarantees, write-back to data sources as well as detection of changes made on data sources bypassing `iMeMex`. 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. Keynote: MyLifeBits: a Memex-Inspired Personal Store; Another TP Database. 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. to appear.

[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. Y. Halevy. Malleable Schemas: A Preliminary Report. In *WebDB*, pages 139–144, 2005.

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

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

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

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

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

[15] 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.

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

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

[18] 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.

[19] D. Kossmann. The State of the Art in Distributed Query Processing. *ACM Computing Surveys*, 32(4):422–469, 2000.

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

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

[22] http://msdn.microsoft.com/data/WinFS/ WinFS.

[23] http://www.imemex.org. iMeMex project web-site.