# A Dataspace Odyssey: The iMeMex Personal Dataspace Management System*

Lukas Blunschi Jens-Peter Dittrich Olivier René Girard
Shant Kirakos Karakashian Marcos Antonio Vaz Salles

ETH Zurich, Switzerland
dbis.ethz.ch | iMeMex.org

## ABSTRACT

A Personal Dataspace includes all data pertaining to a user on all his local disks and on remote servers such as network drives, email and web servers. This data is represented by a heterogeneous mix of files, emails, bookmarks, music, pictures, calendar, personal information streams and so on. We demonstrate a new breed of system that is able to handle the entire Personal Dataspace of a user. Our system, named iMeMex (integrated memex), is a first implementation of a *Personal DataSpace Management System (PDSMS)*. Visions for this type of systems have been proposed recently [13, 10, 12, 17]. We showcase how iMeMex allows dataspace navigation *across data source/file boundaries*, how iMeMex offers *rich contextual information* on query results and how our system returns *best-effort results*.

## 1. INTRODUCTION

In 1945, Bush [6] presented a vision of a personal information management (PIM) 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 functionalities [18, 5, 8, 21]. In addition, PIM regained interest in the Database research community [20, 14]. Moreover, it was identified as an important topic in the Lowell Report [1], discussed in a VLDB panel [22], and became topic of both SIGMOD 2005 keynotes [5, 24] and of a SIGIR workshop [26].

**Personal Information Jungle.** We argue that a satisfactory solution has not yet been brought forward to many central issues related to PIM. In fact, today's users are faced with a jungle of data processing solutions and a jungle of data and file formats [13]. This illustrates two key problems in the current state-of-the-art for PIM: *physical* and *logical data dependence* for personal information. *Physical data dependence* relates to the fact that users need to know about devices and formats that are used to store their data. *Logical data dependence* relates to the fact that users cannot define user-centric views over the data model that is used to represent data. Currently, there is no single system capable of offering an abstraction layer that overcomes both problems and enables data processing (e.g., querying, updating, performing backup and recovery operations) across files, formats and devices.

**From Databases to Dataspaces.** DBMS technology successfully resolved the physical and logical data dependence problem for structured data, but not for the highly heterogeneous data mix present in personal information. Franklin et al. [17] recognize this situation for a variety of domains, including PIM, and present a broad vision for data management. They argue for a new system abstraction, a *DataSpace Support Platform (DSSP)*, capable of managing all data of a particular organization, regardless of its format and location. Unlike standard data integration systems, a DSSP does not require expensive *semantic data integration* before any data services are provided. For example, keyword searches should be supported at any time on all data (*schema later* or *schema never*). A DSSP is a *data co-existence approach* in which tighter integration is performed in a *"pay-as-you-go"* fashion [17].

**From Vision to Reality.** In this demo, we focus on *personal dataspaces*, that is the total of all personal information pertaining to a certain person. In contrast to the vision of [17], we propose a concrete *Personal Dataspace Management System (PDSMS)* implementation, named iMeMex: integrated memex [10, 9]. A PDSMS can be seen as a specialized DSSP. Note, however, that our system is not restricted to personal dataspaces and can also be applied to other scenarios, e.g., scientific dataspaces. A first prototype of our system was demonstrated in [13]. After that, we developed a unified data model for personal dataspaces [12]. This demo presents the second prototype of iMeMex which is based on the findings of [12]. Furthermore, we make the following **contributions**:

1. We summarize the vision of iMeMex. In addition, we list current and upcoming features of our system.
2. iMeMex frees the data contained in a dataspace from its formats and devices, by representing it using a logical graph model [12]. We present how to navigate, search and query a dataspace managed by iMeMex using our AJAX interface.
3. We showcase how iMeMex allows to progressively return best-effort results for a dataspace scenario consisting of distributed iMeMex instances.

This paper is organized as follows. The next section outlines the vision of our system. After that, we summarize iMeMex's architecture in Section 3. In Section 4, we list the current features of our software which is open source and available under an Apache 2.0 License from [9]. Section 5 presents the demonstration outline. Sections 6 discusses related work. Section 7 concludes the paper.

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


3rd Biennial Conference on Innovative Data Systems Research (CIDR) January 7-10, 2007, Asilomar, California, USA.

## 2. THE IMEMEX VISION

Some aspects of our vision were already presented in [13] and [10]. This section summarizes the core ideas. The ultimate goal of the iMeMex project is to free users from logical and physical data management concerns. What this means for a user is discussed in the following paragraphs:

**PIM today:** Assume that Mr. John Average owns a set of devices including a laptop, a desktop, a cellular, and a digital camera. His personal files are spread out among those devices and include music, pictures, pdf files, emails, and office documents. Today, Mr. Average has to copy files from one device to the other, he has to download data to his desktop to see the pictures he shot, he has to upload pictures to sites like *flickr.com* or *picasa.com* to share them with his family. He has to make sure to regularly backup data from the different devices in order to be able to retrieve them in case of a device failure. Further, he uses two different modes of searching: a local desktop search engine enabling search on his local devices and a web search engine enabling search on public web-sites. Mr. Average may organize his files by placing them in folder hierarchies. However, the files and data items stored on his different devices are not related to each other.

**iMeMex vision of 2010:** Mr. John Average still owns several nifty devices with growing processing and storage capabilities. Instead of handling 'devices' he assigns all his data to a logical *dataspace* named John's space. Whenever he listens to a piece of music, takes a picture, gets an email, etc., those items are assigned to John's space. His dataspace management system takes care of the low-level issues including replicating data among devices, enabling search and querying across devices. Whenever John Average wants to share data, he simply creates a subdataspace like John's space:pictures and selects a list of people who may see that data, e.g., his family or friends. There is no need to 'upload' or 'download' data: John's family and friends will just *see* John's pictures without requiring to access web-servers or messing around with files. The boundary between the Web and the different operating systems running on his local devices is gone. However, John Average still *owns* his data: all master copies of his data are physically stored on devices he owns. Searching a dataspace is not restricted to certain devices (like the local desktop), but includes all devices containing data assigned to his dataspace. Other than simple keyword queries, structural queries similar to NEXI [28] are enabled. John Average may also search and query the dataspaces of his friends and his family. The search granularity is fine-granular 'resource views' [12] and *not* files. Other than just searching or querying, John Average may also use iMeMex to integrate the information available in his dataspace or his friends' dataspaces in a pay-as-you-go fashion. Therefore, his dataspace management system analyzes the data and proposes relationships among data items. It enhances his dataspace over time and helps to turn a set of unrelated data items into integrated information. Finally, Mr. Average may also update data using iMeMex. However, he may still update his data using any of his applications, bypassing iMeMex.

In this demo we show two aspects of the iMeMex Dataspace Management System: (1) how to navigate and query a dataspace, (2) how to handle best-effort results computed by distributed instances of iMeMex.

## 3. iMeMex CORE ARCHITECTURE

In this section, we discuss the core architecture of the iMeMex PDSMS. iMeMex is based on a layered architecture which is described in Section 3.1. Following that, Sections 3.2 and 3.3 discuss important services that are provided by the different layers.

### 3.1 Logical Layers

The DSSP vision of Franklin et.al. [17] defines a dataspace as a set of participants (or data sources) and relationships among the participants. We term the set of data sources *Data Source Layer*. Although Franklin et al. [17] present services that should be provided by a DSSP, little is said on how a DSSP would provide those services on top of the Data Source Layer. In fact, in the current state-of-the-art for personal information management, applications (e.g., search&browse, email, Office tools, etc) access the Data Source Layer (e.g., file systems) directly. This comes at the cost of physical data dependence, as for instance system dependence. This situation is depicted on the left of Figure 1.

To remedy that situation, we argue that what is missing is a logical layer between the applications and the Data Source Layer that provides services on the dataspace. We propose to add the iMeMex PDSMS as that intermediate logical layer. It is depicted on the right of Figure 1. iMeMex abstracts from the underlying subsystems, from data formats, and from devices, providing a coherent view to all applications. iMeMex, however, does not have full control of the data as it is the case with DBMSs. Thus, applications may also access the data sources bypassing iMeMex, e.g., email or office applications do not have to be rewritten to interact with iMeMex: they work directly with the data sources. Other applications, however, may be rewritten to directly operate on iMeMex, e.g., explorer and tcsh[1].

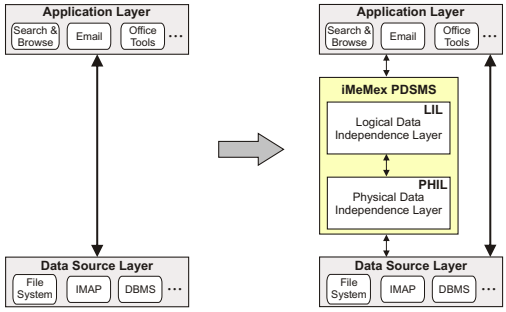


**Figure 1: iMeMex remedies the current state-of-the-art in PIM by introducing logical layers that abstract from underlying subsystems, from data formats, and from devices.**

In the following, we discuss the characteristics of each layer of the iMeMex PDSMS as well as of the layers with which it interacts. All of these layers are shown on the right of Figure 1.

**Data Source Layer.** This layer represents all subsystems managed by the PDSMS. A subsystem that participates on the dataspace may offer either an API that enables full access to the data on that subsystem, access through querying only, or a hybrid of these two options. Thus, the PDSMS must be aware of data vs. query shipping trade-offs [23] to enable efficient query processing.

**Physical Data Independence Layer (PHIL).** This layer is responsible for resolving the data model, access protocol, and format dependence existing on the data sources participating in the dataspace. PHIL offers unified services such as *data model integration* and *indexing and replication*. We provide more details on these services in Section 3.2.

**Logical Data Independence Layer (LIL).** This layer provides view definition and query processing capabilities on top of PHIL. LIL offers services such as *result caching*, *view materialization* and *dataspace navigation* for views defined on top of the data unified

[1]Another approach was taken in [13] where we exposed the dataspace managed by iMeMex through a network interface which could then be mounted by existing operating systems.

by PHIL. We discuss important aspects of these services in Section 3.3.

**Application Layer.** This layer represents the applications built on top of the iMeMex PDSMS. As a PDSMS does not obtain full control of the data, applications may choose to either benefit from the services offered by the PDSMS or access the underlying data sources and use specialized APIs. To enable legacy applications to directly interface with the PDSMS, a PDSMS may offer a mechanism for integrating seamlessly into the host operating system, as demonstrated in [13].

## 3.2 PHIL Services

The primary goal of PHIL is to provide physical data independence. Thus, PHIL unifies data reachable in distinct physical storage devices, access protocols and data formats. We present the main services offered by PHIL below.

**Data Model Integration.** Data model integration refers to the representation of all data available in the data source specific data models using a common model: the *iMeMex Data Model (iDM)* [12]. In a nutshell, iDM represents each piece of personal information by fine-grained logical entities. These entities may describe files, structural elements inside files, tuples, data streams, XML, or any other piece of information available on the data sources. These logical entities are linked together in a graph that represents the entire personal dataspace of a given user. The details of our data model are beyond the scope of this paper (please see [12]). In the remainder of this paper we use the terms **resource view** and **resource view graph** to refer to a logical piece of information, and a graph of logical pieces of information, respectively. The iMeMex approach is in sharp contrast to *semantic integration*, in which expensive up-front investments have to be made in schema mapping, in order to make the system useful. We follow a pay-as-you-go philosophy [17], offering basic services on the whole dataspace regardless of how semantically integrated the data is. We are currently developing a powerful framework for pay-as-you-go integration on top of our data model. This, however, is not the focus of this demo.

**Indexing and Replication.** Given a logical data model to represent all of one's personal information, the next research challenge is how to support efficient querying of that representation. One may consider a pure mediation approach, in which all queries are decomposed and pushed down to the data sources. Though this strategy may be acceptable for local data sources, it may incur long delays when remote data sources are considered. In order to offer maximum flexibility, PHIL offers a hybrid approach. Our approach is based on a *tunable mechanism to bridge warehousing and mediation*. For example, we may choose to replicate relationships among resource views that come from remote data sources, but neither index nor replicate their content. In this situation, relationship navigation among resource views can be accelerated by efficient local access to the replica structures, while retrieval of resource view content will incur costly access to (possibly remote) data sources.

## 3.3 LIL Services

The primary goal of LIL is to provide logical data independence. LIL enables posing complex queries on the resource view graph [12] offered by PHIL. We discuss the services provided by LIL in the following paragraph.

**Personal Dataspace Search&Query Language.** LIL processes expressions written in a new search&query language for schema-agnostic querying of a resource view graph: the *iMeMex Query Language (iQL)*. 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, different from 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 semantic integration. Therefore, as in content and structure search languages (e.g. NEXI [28]), our goal is to account for impreciseness in query semantics. For example, by default, when an attribute name is specified (e.g. size>10K), we should not require exact matches on the (implicit or explicit) schema for that attribute, but rather return fuzzy, ranked results that best match the specified conditions (e.g. size, fileSize, docSize). This allows us to define malleable schemas as in [15]. A PDSMS, however, is not restricted to search. Other important features of iQL are the definition of extensible algebraic operations such as joins and grouping (see [12]).

**Result Caching.** The caching of query results is used to speed up the computation of views. iMeMex's approach to query processing is based on lazy evaluation: whenever matching results are present in the Data Source Layer, PHIL, and/or LIL, then these results are retrieved from the highest of those layers. However, in this scenario, the freshness of the data may be lower at higher levels in the query processing stack. As a consequence, query processing must take QoS concerns (e.g., freshness) into consideration. Our strategy is to deliver stale results quickly and then update the result list as fresh data is delivered from the data sources (see Section 5.3).

**Dataspace Navigation.** Users of information systems typically do not start with a precise query specification, but rather develop one in the course of querying and observing results. We call the process of refining query conditions based on a previous definition of the query *dataspace navigation*. It is a common pattern in the exploration of personal information but also data warehousing [11]. In general, if any given set of views were previously computed and had their results cached at LIL, the research challenge is to detect whether a new query may be answered using those views [19]. In difference to [19], these techniques have to work on arbitray content represented as a resource view graph [12].

# 4. SYSTEM FEATURES

## 4.1 Current Features

In this section we present current features of our system as of December 2006.

1. The server is implemented in Java 5 and is platform independent. It currently consists of 50,000 Lines of Code and 536 classes.
2. iMeMex is based on a service-oriented architecture as defined by the OSGi framework (similar to Eclipse). This means that services, e.g., data source plugins or content converters, can be exchanged at runtime. Our server may be run with two different OSGi implementations: Equinox [16] or Oscar [25].
3. All data is represented by the *iMeMex Data Model* [12].
4. Our query parser supports an initial version of iQL as presented in [12]. iQL supports a mix of keyword and structural search expressions.
5. We provide a rule based query processor that is able to operate in three different querying modes: warehousing (only local indexes and replicas are queried), mediation (local indexes are ignored, queries are shipped to the data sources), and hybrid (combination of the former methods).
6. We provide several different indexing strategies implemented on top of a relational database and a full-text search engine. The relational portions of resource views are vertically decomposed [7, 2] to provide better response times. Our primary target is to develop indexes that operate on external memory.

However, some of our index structures are main memory resident. Which indexes to use is fully configurable.

7. Scalability: our current version is able to handle up to 25 GB of indexed data (net size, excluding image or music content) on a single iMeMex instance. The biggest file indexed was 7 GB.
8. We have implemented wrapper plugins for the following data sources:
    1. File systems (platform independent: works for Windows, Linux and MAC OS X)
    2. Network shares (SMB)
    3. Email server (IMAP)
    4. Databases (JDBC)
    5. Web documents (RSS/ATOM, i.e., any XML data that is accessible by a URI)
9. We provide content converters for LaTeX, Bibtex, XML (SAX-based), and PDF.
10. iMeMex provides two important interfaces:
    1. A text console that allows to perform all administration tasks and also allows to query the server.
    2. An HTTP server supporting three different data delivery modes: HTML, XML, and binary. We are also developing an iMeMex client that accesses the iMeMex server through the HTTP interface. The current version of that client is shown in the demo.
11. The iMeMex server is open source (Apache 2.0 License) since December 2006 and available at http://www.imemex.org.

## 4.2 Upcoming Features

We are planning to provide the following features with upcoming releases of our software:

1. Pay-as-you-go information integration based on a new declarative framework
2. OS integration for file events (Mac and Windows, using native libraries and C++)
3. Cost-based query optimization
4. Integration of updates from data sources
5. Data replication and sharing framework
6. Support for larger datasets > 25 GB, scaling beyond 1 TB using distributed instances

# 5. DEMO OUTLINE

Our demo consists of three parts. First, we introduce our AJAX GUI built on top of iMeMex (Section 5.1). Second, we present how to navigate the personal dataspace using that client and our query language iQL (Section 5.2). Third, we present a use-case that illustrates best-effort results (Section 5.3).

## 5.1 iMeMex Dataspace Navigator

In this Section we introduce the iMeMex Dataspace Navigator GUI. It is displayed in Figure 2. Our GUI consists of three main components: (1) the iQL Search&Query Box, (2) the iDM Graph Display, and (3) the Result Display.

The **iQL Search&Query Box** can be found in the upper part of the GUI. This box can be used to enter arbitrary iQL expressions. iQL is the search&query language we use to query the iDM graph. In contrast to search engine approaches (e.g., Google), operating systems (e.g., Windows Explorer), and DBMSs (e.g., languages such as SQL and XQuery), our iQL language unifies keyword search, structural querying, addresses and paths into a single language. iQL may be used by non-experts and experts. Please see [12] for details.

The **iDM Graph Display** occupies the left part of the GUI. It displays the dataspace managed by iMeMex. Currently, we use a tree-like representation to represent that graph (i.e., we show all possible paths on the graph starting from nodes chosen by the user). The reason is that today's users are already familiar with tree-like interfaces. This way only little learning effort is required to switch from managing traditional files&folders to managing a logical dataspace. However, we are also planning to explore other visualizations.

The **Result Display** occupies the biggest area of the GUI. It is found in the middle and right of Figure 2. It shows the results returned by the iQL query typed on the Search&Query box or clicked in the iDM Graph Display. For each result, we show its name, its properties (e.g., from and to for emails), and a URI that locates the result in its underlying data source. In contrast to standard desktop search engines, we also find links to rich contextual information associated to that result. We describe how the user may benefit from that contextual information in the following.

## 5.2 Use Case 1: Navigating the Dataspace

We present a typical iMeMex session performed by a user. The user may begin her exploration of the dataspace by navigating through the iDM Graph Display (left of Figure 2). That display unifies data from several data sources into one single browsing interface. In the example of Figure 2, we show data coming from the file system and from an email server. Further, the iDM Graph Display not only allows the user to browse files&folders, but also the structural information inside the files. In Figure 2, we display the document structure for the file CIDR 2007.tex present in the papers folder in the filesystem. Note that the same functionality is available across data sources (e.g., email file attachment introduction.tex in Figure 2). When a user selects a node in the iDM Graph Display, details for that node are displayed in the Result Display.

We proceed by showing how a user may submit an iQL query to the system. Let's assume the user wants to start by entering a simple keyword such as CIDR. That keyword may be entered in the iQL Search&Query Box. Note that a typical user will only enter keywords in that box. Advanced users, however, may enter more complex expressions that restrict querying on certain subgraphs or attributes. For instance, the following query could also be entered in that box:

```
//ETH//*["dwh" and from~"Donald Kossmann"]
```

This query returns all resources that can be reached by graph traversal via a node named ETH, i.e., we generalize the *subfolder* relationship. Further, all results should preferably contain the keyword dwh and have an attribute from similar to Donald Kossmann. Note that future versions of iQL will also allow to specify algebraic operations such as joins and aggregations. The development of iQL is ongoing work [12].

When the user presses the Find button, the Result Display shows the top-k results that were computed by iMeMex. For each result shown, the user may benefit from a rich source of contextual information which includes:

**Graph connections.** In the iDM Graph, a given query result has a set of incoming and a set of outgoing connections (edges) to other nodes. These nodes form the *neighborhood* of that query result. Navigating the neighborhood is useful when the user knows that the result she is looking for is 'related' to another query result, e.g., "what is the name of the person that sent me Donald Kossmann's Data Warehouses lecture notes?".

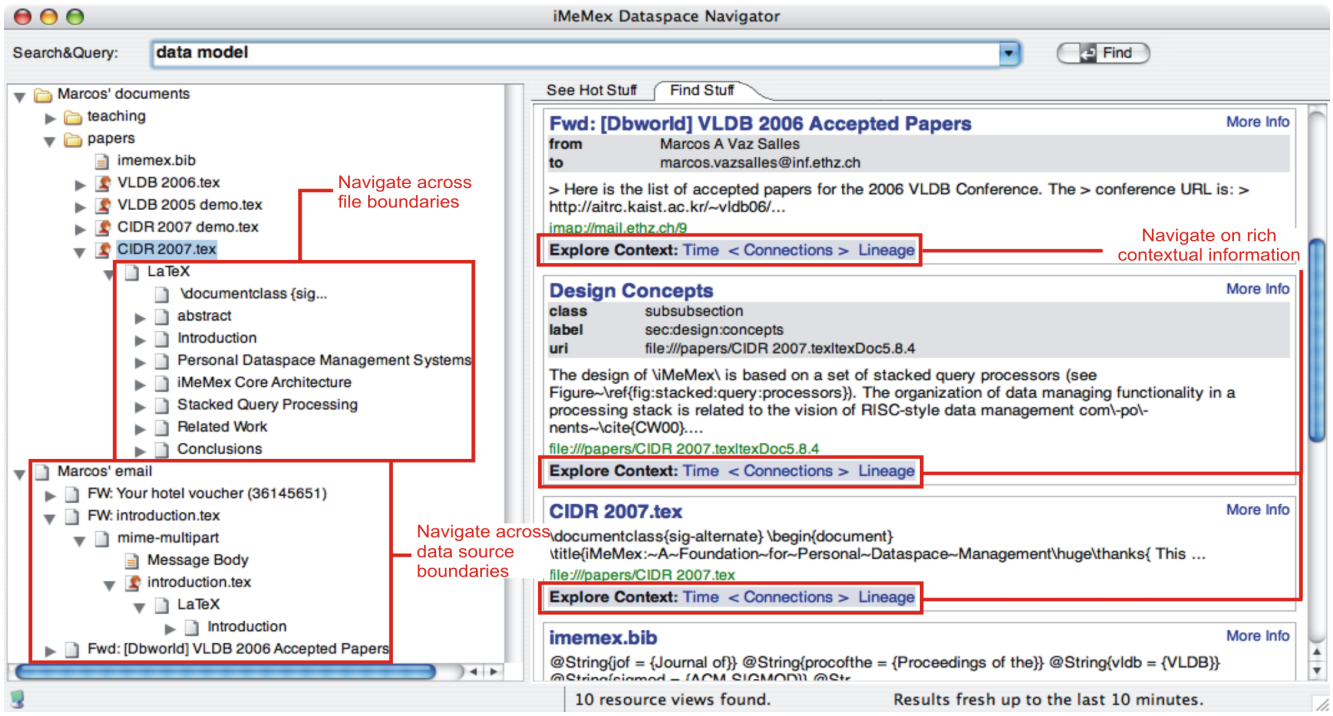


**Figure 2: The `iMeMex` Dataspace Navigator displays a global view of the user's personal dataspace. It enables users to query that global view, to navigate on rich contextual information and to visualize progress and freshness indicators for query results.**

**Time.** The time context is useful to find resources "that were touched about the same time as the query result". Note that not only time, but any other ordered attribute recognized by an application may be used to display nodes that are *near* a given query result. Geographic locations could be used to attach information to maps, e.g., assigning pictures from the user's last holiday to his travel itinerary. This concept is similar to a drill-down in OLAP.

**Lineage.** Lineage refers to the history of data transformations that originated a given node. Users may be interested in obtaining previous versions of a given query result, e.g., to see how a project proposal looked like one month ago. Furthermore, it may also be interesting to understand how a node was created. If a node *a* was copied from a node *b*, then previous versions of *b* may be of interest. For example, an error in a project proposal *a* may have been caused by an error in the proposal's template *b*.

## 5.3 Use Case 2: Best-Effort Querying of the Dataspace

To provide interactive response times for queries on the personal dataspace, `iMeMex` may replicate and index data from the underlying data sources. This implies that the latest updates performed directly on the data sources bypassing `iMeMex` might not yet be reflected in those replicas and indexes. That is why `iMeMex`'s query processor may chose to produce *best-effort* query plans, i.e., query plans that first return low cost/stale results and then complement those results by fresh/high cost ones fetched directly from the data sources. This implies that the initial results displayed by the client will be updated as soon as fresher results are delivered by the underlying system. This is similar to the result update feature of Apple Spotlight [3]. However, in contrast to the latter, our system is much more than a mere search engine.

The `iMeMex` Dataspace Navigator GUI allows the user to visualize `iMeMex`'s best-effort query processing functionality. Consider that the keywords `data model` are submitted by the user. Figure 2 displays the `iMeMex` Dataspace Navigator GUI, while processing this query. On the lower right corner of the figure, we see the result freshness indicator. It states that all data modified more than 10 minutes ago has already been considered for the current query and that the corresponding matches are included in the results. The status bar also indicates that 10 results were found so far. At this point, the user may decide to wait until fresher results appear, or may decide to continue navigating through the returned results.

# 6. RELATED WORK

The abstraction of *personal dataspaces* calls for a new kind of system that is able to support the entire personal dataspace of a user. We term this kind of system a *Personal DataSpace Management System (PDSMS)*. So far, no such system exists. Due to space constraints, we only briefly mention a few related solutions in this section. We focus on positioning these solutions in an overview of the design space for PDSMSs. We refer the reader to [12] for a detailed discussion on related existing approaches.

Figure 3 displays the design space of existing solutions for information management. The horizontal axis displays requirements for semantic integration, while the vertical axis, in contrast to [17], displays the degree of update guarantees provided by different systems. One crucial aspect of dataspace management systems is their need to provide a pay-as-you-go information integration framework that allows to integrate data without defining a global schema. For this reason, a PDSMS occupies the design space in-between the two extremes 'high semantic integration' (schema-first) and 'low semantic integration' (no schema). See also [17].

On the lower left corner of the mentioned design space we find DBMSs, which require high semantic integration efforts (upfront investment for schemas), but provide strong update guarantees (ACID). Examples of systems that attempted to apply DBMS technology to personal information include WinFS [29], MyLifeBits [5] and Rufus [27]. These solutions, however, incur high costs for semantic

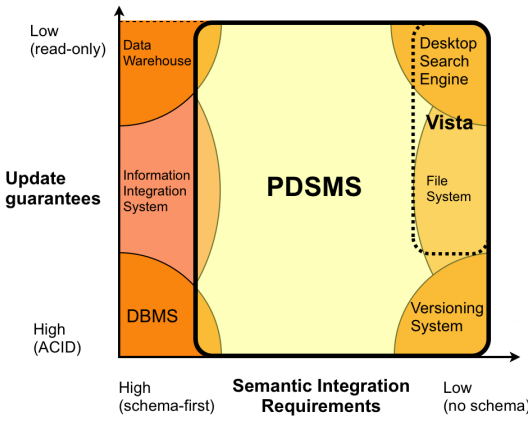


**Figure 3: Design space of state-of-the-art information management systems: PDSMSs fill the gap between existing specialized systems.**

integration and require full control of the data.

Strictly opposed to that, a desktop search engine (DSE) does neither require semantic integration, nor full control of the data. On the other hand, these systems do not provide any update guarantees and do not allow structural information to be exploited for queries. Examples of such systems are Google Desktop, Apple Spotlight, Beagle [4], and Phlat [8]. The upper left corner of Figure 3 is occupied by data warehouses: these systems are optimized for read-only access. Furthermore, they require very high semantic integration efforts (integration of multiple schemas). Figure 3 also shows traditional information integration systems (middle-left): these systems require high semantic integration investments and vary in terms of their update guarantees. Some systems, such as SEMEX [14] and Haystack [21], extend data warehouse and information integration technology. They 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 global schema over the personal information sources. This schema-first approach makes it hard to integrate information in a pay-as-you-go fashion as required by a dataspace management system. In fact, all of the mentioned systems may be considered as applications on top of a data managing platform such as the `iMeMex` PDSMS.

The lower right corner of Figure 3 is occupied by versioning systems (e.g., Subversion, Perforce), which provide strong update guarantees but do not require semantic integration. File systems occupy the region on the middle-right, providing weaker update guarantees than versioning systems (e.g., recovery on metadata for journaling file systems). The upcoming operating system Windows Vista is also displayed as it provides some basic information management capabilities (dotted box on the upper right corner), covering functionalities offered by file systems and DSEs.

Figure 3 shows that a huge design space between the different extremes (sitting in the corners and along the margins) is not covered by current information management solutions. However, in order to be able to manage the entire dataspace of a user that space has to be covered. PDSMSs fill that gap. These systems cover the entire design space of information systems requiring medium to low semantic integration efforts. Furthermore, PDSMSs occupy the middleground between a read-only DSE (without any update guarantees) and a write-optimized DBMS (with strict ACID guarantees).

# 7. CONCLUSIONS

This paper has advocated the design of a single system to master the personal information jungle [13]. The `iMeMex` Personal Dataspace Management System introduces a logical layer on top of the data sources that provides full physical and logical personal information independence. Our PDSMS is based on the OSGi framework and thus can be extended at (almost) any granularity. We have shown how `iMeMex` can be used by a search&browse GUI client to provide the user with a global view of her personal dataspace. For that purpose, we have detailed two use cases: the first illustrated how our GUI allows users to navigate and query their entire dataspace. The second showed how best-effort query results are provided. As future work, we plan to provide the upcoming features as listed in Section 4.2.

# 8. 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] R. Agrawal, A. Somani, and Y. Xu. Storage and Querying of E-Commerce Data. In *VLDB*, 2001.

[3] `http://www.apple.com/macosx/features/spotlight/` Apple Mac OS X Spotlight.

[4] `http://beaglewiki.org/` Beagle.

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

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

[7] G. P. Copeland and S. Khoshafian. A Decomposition Storage Model. In *ACM SIGMOD*, pages 268–279, 1985.

[8] E. Cutrell et al. Fast, flexible filtering with Phlat — Personal search and organization made easy. In *CHI*, 2006.

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

[10] J.-P. Dittrich. iMeMex: A Platform for Personal Dataspace Management. In *SIGIR PIM Workshop*, 2006.

[11] J.-P. Dittrich, D. Kossmann, and A. Kreutz. Bridging the Gap between OLAP and SQL. In *VLDB*, 2005.

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

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

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

[15] X. Dong and A. Y. Halevy. Malleable Schemas: A Preliminary Report. In *WebDB*, 2005.

[16] `http://www.eclipse.org/equinox/` Equinox: Eclipse OSGI implementation.

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

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

[19] A. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

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

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

[22] 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.

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

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

[25] `http://oscar.objectweb.org/` Oscar: OSGi implementation.

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

[27] K. A. Shoens et al. The Rufus System: Information Organization for Semi-Structured Data. In *VLDB*, 1993.

[28] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In *INEX Workshop*, 2004.

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