# iMeMex: Escapes from the Personal Information Jungle

Jens-Peter Dittrich Marcos Antonio Vaz Salles Donald Kossmann Lukas Blunschi

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

## Abstract

Modern computer work stations provide thousands of applications that store data in >100.000 files on the file system of the underlying OS. To handle these files data processing logic is reinvented inside each application. This results in a jungle of data processing solutions and a jungle of data and file formats. For a user, it is extremely hard to manage information in this jungle. Most of all it is impossible to use data distributed among different files and formats for combined queries, e.g., join and union operations. To solve the problems arising from file based data management, we present a software system called `iMeMex` as a unified solution to personal information management and integration. `iMeMex` is designed to integrate seamlessly into existing operating systems like Windows, Linux and Mac OS X. Our system enables existing applications to gradually dispose file based storage. By using `iMeMex` modern operating systems are enabled to make use of sophisticated DBMS, IR and data integration technologies. The seamless integration of `iMeMex` into existing operating systems enables new applications that provide concepts of data storage and analysis unseen before.

## 1 Introduction

Modern computer workstations like PCs and Macintosh computers provide a myriad of different applications. Each of these applications processes data in some way and stores that data in files on the file system of the underlying operating system.



**Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005**

**Example (Stock Data):** A text fragment may contain unstructured information like in the following example:

```
Warehouse Z contains 56 boxes of product X
 as well as 45 boxes of product Y.
```

This kind of unstructured information is easy to read for a human reader. For a machine, however, the semantics of the above sentence is hard to extract. For this reason, information systems store data in a structured format wherever possible. The following text fragment contains the same information in a structured format:

```
#stock of warehouse Z
#product, stock
X, 56
Y, 45
```

Unfortunately, not every text fragment can be transformed into a structured representation. For this reason, XML was created as a means to structure a document at least partially. This *semi-structured data* may represent table-like as well as hierarchical data:

```
<stock>
  <product>
    <name> X </name>
    <stock> 56 </stock>
  </product>
  <product>
    <name> Y </name>
    <stock> 45 </stock>
  </product>
</stock>
```

Obviously, XML documents contain a lot of redundant information. Nevertheless, the main advantage of this format is the fact that it can be generated and processed by a growing number of applications.

Another method for storing textual data is to make use of *binary file formats*. E.g., Office applications like Word and PowerPoint store data using proprietary file formats. These files contain the actual data, meta data, formatting instructions and undo information. The drawback of using binary files is that these formats can only be processed by a small number of associated applications. This inhibits data exchange and integration with other applications.

### 1.1 Problem Statement

Desktop computers use file based storage to store information. A typical desktop computer contains hundreds of thousands of files[1] using hundreds of different file formats. This file based storage leads to a variety of problems:

1. No or difficult data exchange among different applications
2. No central service that manages data on the desktop and provides query services on that data
3. No possibility to join data stored in heterogeneous files
4. No automatic recovery of application data in case of power, hard disk or other hardware failures
5. No support for automated backup of data
6. No support for automated versioning and replication of data
7. No support for concurrent access to files or fragments of files
8. No support for personalized, context-aware search nor filtering

The major reason for this unsatisfying situation is due to the fact that each application reimplements its own data processing algorithms. In many cases, these algorithms are just simple variants of long-matured DBMS algorithms. Algorithms that have been shipped with DBMS for decades.

### 1.2 Consequences

In summary, when trying to manage personal information on a desktop computer, today's users are confronted with the following problems:

1. A desktop computer is a jungle of information and data processing solutions.
2. A desktop computer is a jungle of unsatisfying solutions for recovery, backup, versioning and synchronization.
3. It is hard to find and store information in this jungle.

## 2 Related Work

### 2.1 Related Literature

Personal Information Management (PIM) has recently been identified to be of high interest for DB research [5, 1]. Kersten et.al. [5] state that Mister Average is nowadays confronted with a dozen of databases of different formats. Currently, it is not even possible to perform simple joins among those different databases. Therefore, the authors strengthen the need to provide means to integrate that personal data. The Lowell report [1] considers information integration as one of the key problems of modern information systems. The authors state that the distribution of sensors and datasets throughout the world breaks the ETL paradigm traditionally applied by information integration tools. Therefore, new approaches to information integration are required. Recently, Dong et.al. [3] propose a PIM system that provides automatic annotations of data. The proposed system tries to generate semantic associations which can then be browsed by the user. Halevy et.al. [4] claim that most of the world's data lies outside DBMSes. The authors discuss the structure chasm between the structured world of the DBMS and the unstructured world of files.

### 2.2 Why WinFS and Tiger are not enough

There are two approaches to solving the problems described above:

The **first** approach is to *replace the file system of the operating system by a DBMS*. This is the strategy of proposals like WinFS[2]. The core idea of WinFS is to store meta data and data in a relational database whenever possible. However, the WinFS approach has several drawbacks:

1. WinFS is focused on relational data. All data 'items' are mapped to rows in a database. XML data is only treated as a second class citizen.
2. Only if the application developer is willing to provide a schema, application data inside files will be used for query processing.
3. Results are always returned as a list of items. This is not beneficial if a user executes a query that returns a large XML document. That document would then be represented as a single item.
4. WinFS is based on a pull-based RDBMS. Push-based operators, however, which are needed to provide efficient stream processing, are not supported.
5. WinFS does not provide a generic plugin-concept to extend its functionality.
6. WinFS is platform specific.

The **second** approach is to *index all files of a system by a search engine*. Currently, only Mac OS X Tiger ships with a built in search engine. For other platforms add-ons like Google Desktop Search are available. Though search engines perform textual keyword searches very efficiently, they do have some drawbacks. One is that the results provided by the search engine are unstructured information, i.e., a simple list of documents that match. Although documents that contain keywords inside tables are also reported as result documents, the table structure is then neither exploited for the search nor for postprocessing (e.g. ranking) the results. Therefore it is impossible to further process the result list of the search engine by using generic operations like *join* and *union*. Since this integration aspect is not exploited by current search engine technology, a rich source of information is wasted.

## 3 iMeMex: As We May Store

In this Section, we introduce the `iMeMex`[3] system as the unified solution to personal information management. `iMeMex` enables the joint analysis of different kinds of data on a user's desktop computer. In order to achieve this

[1] The personal computers of the authors contain on average more than 700.000 files — not counting network folders.

[2] Note that WinFS will only become available after the release of the new Windows release named 'Longhorn'. The latter is scheduled only for end of 2006.

[3] The name is an abbreviation of *integrated memex* where *memex* refers to the vision presented in [2].

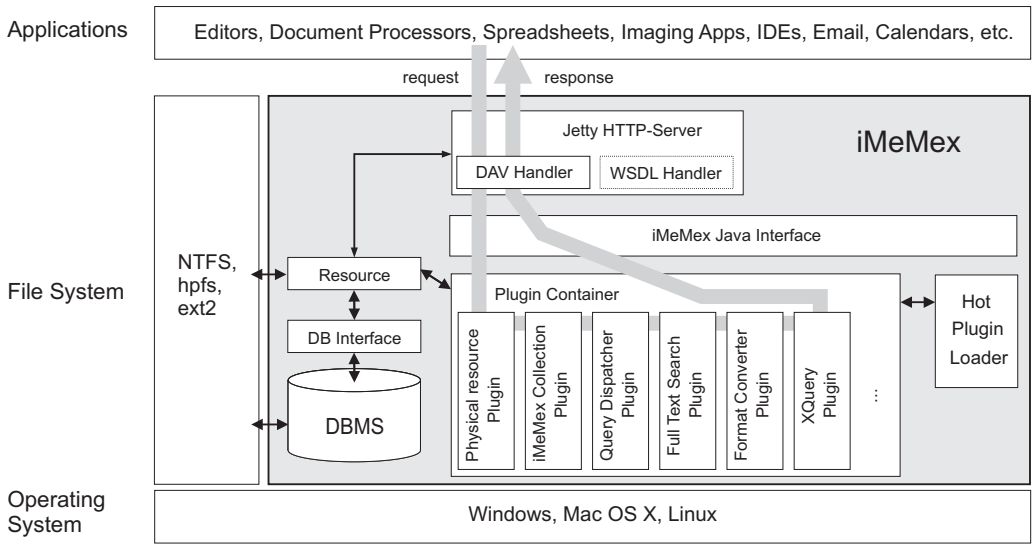



Figure 1: The architecture of the iMeMex personal information management system

goal, iMeMex provides an additional persistence and analysis layer on top of the file system provided by the OS. This persistence and analysis layer provides storage and analysis services that differ fundamentally from todays file based storage. In the long run this layer will replace existing file systems.

### 3.1 OS Integration

Figure 1 gives an overview on the integration of iMeMex into the host operating system. iMeMex resides on the same level as the file systems of the underlying operating system. The entire functionality of iMeMex is exposed to other applications through a WebDAV interface. The reason we choose WebDAV is that this protocol is already supported by a large number of applications (Eclipse, Word, etc.) as well as operating systems (Windows, Linux, Mac OS X). The WebDAV protocol is usually used to mount web-based file repositories on local machines. For the user these repositories then look like ordinary file folders.

iMeMex will also provide such kind of folder view. By replacing the users home directory folder with an iMeMex WebDAV folder the iMeMex system gains full control of the user's files[4]. Like that iMeMex does neither require special file import and export coding nor synchronization rules. The system seamlessly integrates into existing operating systems.

In contrast to ordinary WebDAV folders, the iMeMex folder view provides two kinds of resources[5]: first, resources that were added to the iMeMex repository (*physical resources*), and second, resources that display information generated by the iMeMex system (*virtual resources*). The virtual resources provide meta data, structured views on the physical resources, starting points for search operations, generated links to related content and so on.

### 3.2 Architecture

Figure 1 shows the architecture of the iMeMex system. Our system consists of three major components:

1. **HTTP-Server**: A Jetty HTTP-Server is used to handle DAV-requests. We have extended Jetty by a handler that provides support for WebDAV level 2. A future version of our system will also contain a WSDL handler. The WSDL interface will provide direct application support for storing and querying semi-structured and structured data. In order to access that feature, however, applications would have to be changed.
2. **DBMS:** An off-the-shelf open-source DBMS is used. In addition, the DBMS is used to provide efficient storage, indexing and search on structured data. Our current implementation uses Postgres 8.0.
3. **Plugin Container:** The Plugin Container is the heart of iMeMex. That mechanism can easily be used to extend the functionality of our system. The plugin container registers plugins which may subscribe to incoming messages. Note that plugins are automatically loaded into the system by a Hot Plugin Loader. This means we do not have to restart our system when a new plugin becomes available.

### 3.3 Plugins

iMeMex ships [6] with an inital set of important plugins:

1. **Physical Resource PI:** provides information on available resources (e.g., files, XML twigs or tables)
2. **iMeMex Collection PI:** provides an initial set of virtual resources (like meta data, 'iMeMex' folder, etc.)

[4]Note, that in most cases the iMeMex server will reside on the local desktop computer — just like the native file systems of the OS. For other applications, however, the iMeMex server might be located on a remote machine.

[5]In the following, we assume that files and directories are just special resources. A directory is a collection of resources.

3. **Query Dispatcher PI:** support for query processing and views (e.g., information integration and filtering)
4. **Full Text Search PI:** full text search for all physical resources handled by iMeMex (e.g., desktop search)
5. **Format Converter PI:** format transformation routines (like Excel to XML, etc.)
6. **XQuery PI:** support for XQuery

# 4 Description of the Demo

The demo will present the core architecture of iMeMex. We demonstrate the seamless integration of our system into standard operating systems as well as the benefits of iMeMex in terms of personal information management.

## 4.1 Use-Case: Operating System Integration

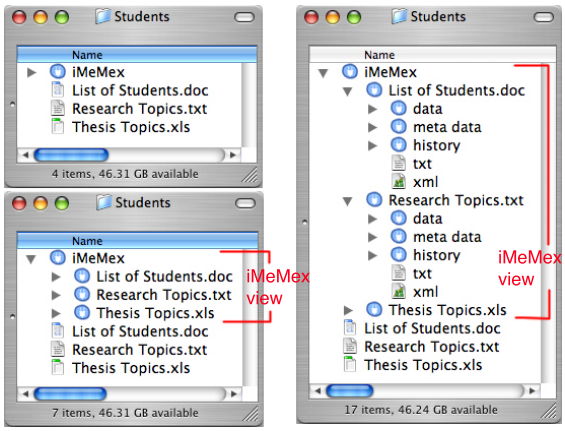
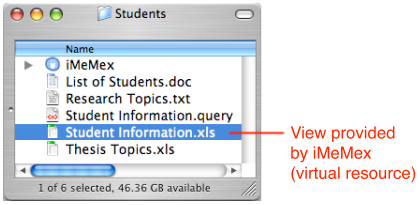



Figure 2: A file view provided by iMeMex

We will demonstrate how iMeMex seamlessly integrates into the host operating system. Figure 2 shows an example OS folder managed by iMeMex. The Figure provides three different views on the same folder *'Students'*. The upper-left window shows that the folder contains three files: *'List of Students.doc'*, *'Research Topics.txt'* and *'Thesis Topics.xls'*. In addition, the *'Students'* folder contains a subfolder *'iMeMex'* which is a virtual resource provided by our system. If we expand that *'iMeMex'* folder we receive the window in the lower left of Figure 2. That window shows that the *'iMeMex'* folder displays a subfolder for each physical file — each of them named like the physical file. If we expand two of these subfolders we receive the window on the right of Figure 2. We see that for each physical file, iMeMex provides a rich source of information: 1. a subfolder providing an abstract folder view on the data, 2. a subfolder containing meta data, and 3. a subfolder containing history and versioning information for that file. In addition, alternative views on the data are provided like: 4. a text view (*txt*), and 5. an XML view (*xml*). Note that the contents of virtual resources are only computed on demand, i.e., if an application or the user accesses that resource. Also note, that the list of virtual resources can easily be extended by deploying appropriate plugins to iMeMex.

## 4.2 Use-Case: Views on the Desktop

We will demonstrate that iMeMex strongly facilitates query processing on heterogeneous information sources. This is achieved by providing views on the desktop. Those views may contain structured, unstructured as well as semi-structured information. We will demonstrate how to use our system to create a view (a virtual resource) on a set of heterogeneous files. The join processing inside that view is performed using XQuery. Note, that we do not materialize the contents of views in advance. A view is only computed if the content of that view gets requested by the user or an application. The view is defined in a `.query` file. That file contains the definition of a view either in XQuery, SQL, or as a keyword search. In addition, the `.query` file defines the output format of a view. For instance, we could define a view named *'Student Information.query'*, defining `excel` as the output format. Then, iMeMex automatically creates a result file named *'Student Information.xls'*:




The content of that file (the result to the query) is only computed when the user tries to read that file.

# 5 Conclusions

A modern desktop computer is a jungle of information and data processing solutions. For a user it is very hard to organize and query information in this jungle. We have presented the iMeMex system as a generic solution to taming the jungle. We have shown that iMeMex seamlessly integrates into existing desktop operating systems enabling them to make use of sophisticated DBMS, IR and information integration technology. As part of future work, we plan to extend iMeMex to integrate web content (like RSS feeds) and streams. In addition, we plan to study P2P networks of iMeMex instances.

# References


[1] S. Abiteboul, R. Agrawal, P. A. Bernstein, M. J. Carey, and others. The Lowell Database Research Self Assessment. *CoRR*, cs.DB/0310006, 2003.

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

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

[4] A. Y. Halevy, O. Etzioni, A. Doan, Z. G. Ives, J. Madhavan, L. McDowell, and I. Tatarinov. Crossing the Structure Chasm. In *CIDR*, 2003.

[5] M. L. Kersten, G. Weikum, M. J. Franklin, D. A. Keim, A. P. Buchmann, and S. Chaudhuri. A Database Striptease or How to Manage Your Personal Databases. In *VLDB*, 2003.

[6] `http://www.iMeMex.org`. Project web-site. *publicly available download is scheduled for Q4 2005.*