Living with a Lifetime Store

Jim Gemmell, Roger Lueder and Gordon Bell
Microsoft Research

Abstract

Storage trends have brought us to the point where it is affordable to keep a complete digital record of one's life. The MyLifeBits system is designed to store and manage a lifetime's worth of data. To experiment with a lifetime store, we have digitized everything possible from Gordon Bell's life. These are added to his existing digital assets. We also continue to add new digital capture such as web pages, telephone, radio and television. In this paper we explain the key requirements of a lifetime store. We show how typed links and database features are keys to the usefulness of such a system.

Introduction

"A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility" – V. Bush

The MyLifeBits system [7] is designed to store and manage a lifetime's worth of *everything* – at least everything that can be digitized. To experiment with a lifetime store, we have digitized everything possible from Gordon Bell's life: articles, books, cards, CDs, letters, memos, music, papers, photos, pictures, presentations, home movies, videotaped lectures, and voice recordings. To this we have added the digital media from his PC such as digital photos, email, and calendar events. MyLifeBits supports capture, storage, management and retrieval of many media types, and logs as much usage data as possible. Gordon now uses MyLifeBits to record every chat session, copy of every web page visited, and selectively record telephone, TV and radio. In this paper we explain the key requirements of a lifetime store. We show how typed links and database features are keys to the usefulness of such a system.

MyLifeBits is inspired by Memex, a personal store envisioned by Vannevar Bush in 1945 for use by scientists [3]. Memex was to store documents, photos, and audio. Bush proposed that Memex support full-text search, voice/text annotations, and hyperlink creation. In the 1960's, Ted Nelson expanded the vision to include features like versioning, hypertext, and transclusion (explained below) [10,11].However, the technology to make their ideas feasible is only now appearing. In the intervening time, personal computers, which started with very limited storage space for few media types (word processing documents and spreadsheets), have been evolving into large and comprehensive stores.

PCs have traditionally stored files in directory trees. Some systems have allowed full-text search, whether by simply scanning (as in grep) or using an index. Such searches can be saved for easy re-issue [1]. Research into personal storage has taken several directions. Some have advocated a purely time-based UI and storage organization [6]. Others have combined time with a location on the user's desktop [12]. Yet others have relaxed hierarchy, allowing files to have more than one parent [1,5].

Over time, the PC has supported new media such as photos, audio and video. Without text to search for, these media have presented special challenges. Annotating non-text media with text is an obvious solution [1,5,9]. Content analysis can also generate meta-data that can be used for search and browsing. For example, speech to text can be run on audio clips, face recognition can be run on photos, and documents may be classified [8,9].

In the remainder of the paper we will discuss trends in data storage and data sources that impact this field. We then consider the requirements for making a lifetime store useful. Based on these requirements, we discuss the underlying data model, and describe our implementation to date. We close with a discussion of outstanding challenges and our conclusions.

Trends in Data Storage and Sources

"yet if the user inserted 5000 pages of material a day it would take him hundreds of years to fill the repository, so that he can be profligate and enter material freely" – V. Bush

Consumer hard drives are currently in the 80 to 300 GB range. It is conservative to predict terabyte hard drives, even for notebook computers, within several years. Such values place us in the era of virtually unlimited storage that Vannevar Bush foresaw.

We have scanned many paper pages into MyLifeBits in TIF format at about 100 KB per page. Thus, Bush's 5000 pages a day would require 0.5 GB of storage. However, collecting paper at this rate is unlikely. One is more likely to accumulate significant quantities of digital entities such as web pages, email, or digital photos. Suppose that you began keeping:

- 100 email messages a day (5KB each)
- 100 web pages day (50KB each)
- 5 scanned pages a day (100KB each)
- 1 book every 10 days (1 MB each)
- 10 photos per day (400 KB JPEG each)

- 8 hours per day of sound - e.g. telephone, voice annotations, and meeting recordings (8 Kb/s)
- 1 new music CD every 10 days (45 min each at 128 Kb/s)

At this rate, it will take you 5 years to fill up your current 80 GB hard drive. By that time, consumer terabyte hard drives will be shipping. Once you upgrade to a terabyte disk, it will take more than 60 additional years to fill.

There is no reason to suppose that disk technology will stop at one terabyte, but even if it did we can still assume that a terabyte will drop below one hundred dollars in price, making the purchase of a new terabyte every year very affordable. Filling a terabyte in a year turns out to be a real challenge. Table 1 shows that it is virtually impossible to fill a terabyte in a year by looking at photos, reading documents, or recording CD-quality audio. Even 256 Kb/s video shot 24 hours a day, seven days a week will not fill up a terabyte in a year. It takes higher bit-rate video to finally fill the terabyte within a year.

Table 1: Trying to fill a terabyte in a year: for each item, the number of items it takes to fill a terabyte, and the number of items per day to fill a terabyte in a year.

Item	*Per TB*	*Per day*
Photo (400 KB JPEG)	2.7M photos	7354 photos
1 MB Document	1.0M documents	2872 documents
128 Kb/s audio	18.6K hours	51 hours
256 Kb/s video	9.3K hours	26 hours
1.5 Mb/s video	1.6K hours	4 hours

So we see that we have already reached the point where one could cheaply store all the digital content you are likely to view, with the exception of video, and that within a few years video will also be accommodated by buying a terabyte hard drive each year. However, there is no reason to believe that drive technology will stop at one terabyte; there appears to be every prospect of reaching 100 TB drives with space for an entire lifetime store.

Digital representations are less expensive than paper (especially accounting for storage space costs). Indeed, if you value your time, it will usually be more expensive to select and delete an item than to keep it. Digital representations will also be easier to backup and geographically distribute, offering better protection against loss due to theft or disaster.

While we have dwelt on storage capabilities for data we are already familiar with, we can also see that data will be captured in less familiar ways. Consider, for example, commercially available on-body systems that joggers use to track distance and speed, or arm-bands that track body functions such as heart rate and temperature. Homes will become more instrumented, enabling conditions (e.g. temperature) along with machine behavior (e.g. when the stove was used, and at what temperature) to be logged. The same can be said for automobiles. In general, we can see that increased sensor logging is to be expected. Of course, cameras and microphones are sensors that will also proliferate and be embedded in many ways. Finally, the user's interaction with input devices and the storage system itself is valuable data to log.

Making the store useful

"A record if it is to be useful … must be continuously extended, it must be stored, and above all it must be consulted" – V. Bush

"The difficulty seems to be, not so much that we publish unduly … but rather that publication has been extended far beyond our present ability to make real use of the record" – V. Bush

Once we see the feasibility of collecting and storing vast amounts of information, the challenge becomes making use of it. There is no point in constructing a "Write Once Read Never" memory. On the other hand, it is not a reasonable requirement that all records will be accessed in the future. In fact, it is common practice for us to keep many records just in case we need to access a single one. We keep many because we cannot predict which one will be useful in the future.

Memex had a number of features to make the store useful. The first was full-text search, whose usefulness has been proven on the World Wide Web and elsewhere. The second was "trails", by which Bush meant the ability to link from one document to the next to guide one through the documents in Memex. Bush's trails were influential to the notion of hyperlinks. However, he concentrated on a single path rather than the web we are all familiar with. One sees this implemented ubiquitously on the WWW as menu bars or table of contents pages, in testimony to its usefulness. Finally, Bush proposed annotations, by voice or handwriting (which would include text).

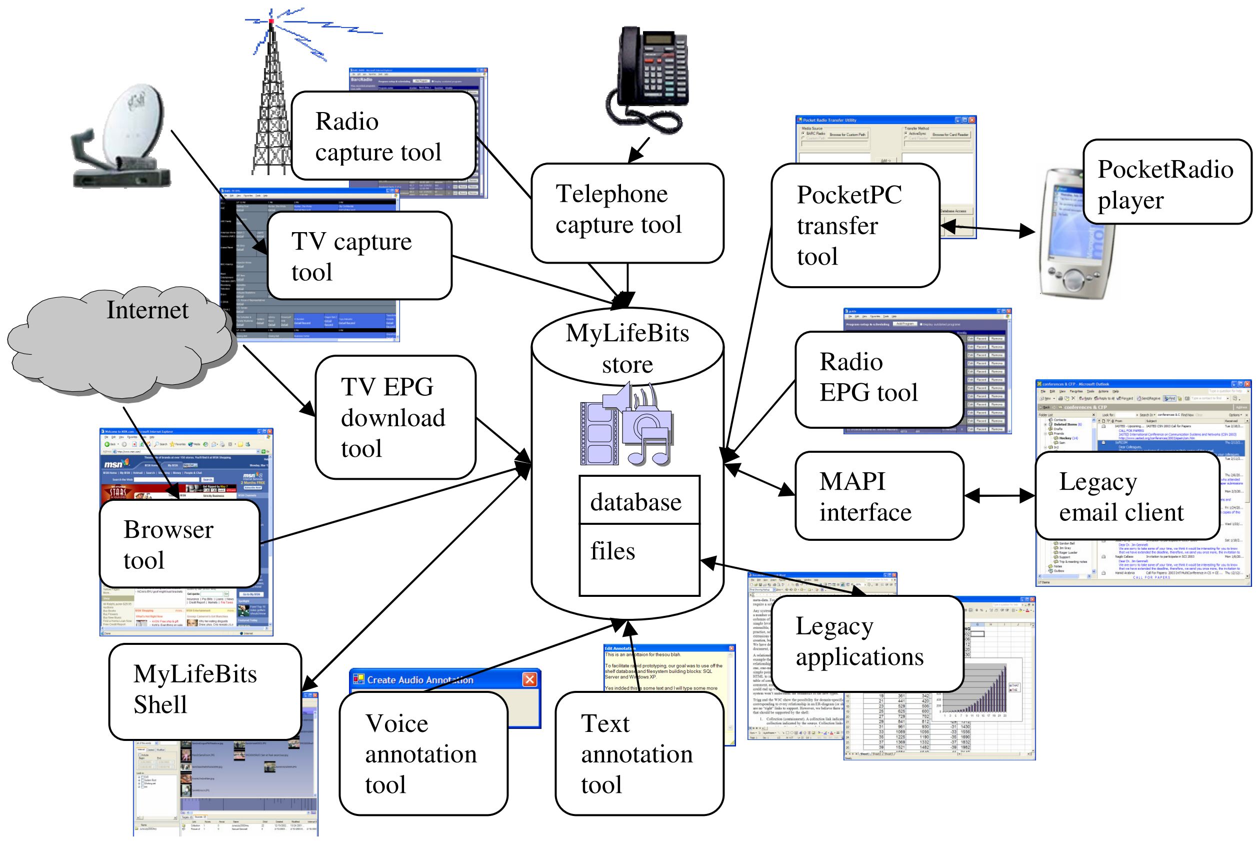


Figure 1 – The MyLifeBits system

In the 1960's, Ted Nelson began extending Bush's ideas [10,11]. His vision was to revolutionize literature in a networked world. Hyperlinks and hypertext were key components to realizing the vision. While Nelson's full vision for literature, with networking and micro-payments, is beyond the scope of our discussion, one concept is of particular usefulness: "transclusion". This was his word for virtually including part or all of one work in another. While the choice of whether to make a copy of the transcluded work may be considered an implementation detail, the critical feature is that links exist from the transcluding work to the transcluded work and vice-versa. A two-way link supports, for example, following a link from a photo to a photo album that it has been transcluded in (to read a caption beside it). Or, if one sees the photo in the album, the link may be followed in the opposite direction to the original photo, where one might find annotations, or that it is in a collection of other photos from the same event that were not included in the album.

It is interesting that while Bush rejected traditional filing in favor of "association", his trails actually facilitate forming collections (each trail being a collection of the items in the trail), and with trails of trails, one can form hierarchies. While strict hierarchy is usually undesirable, (often an item needs to be filed in more than one place) nested collections remain useful. We think that ultimately faceted classifications [2] will subsume most collections, but for now we will simply note that nested collections without strict hierarchy are an important requirement.

Having a single generic link type is too limiting. For example, a link from a contact to a photo should distinguish if the link represents the person being in the photo, or the person being the photographer. We will cover typed links in the discussion of our data model.

Database features also make the store more useful. For example, queries may be saved for repeated execution. These queries may be simple text searches that the user has performed and wants to repeat, or they may involve complex formulae. Also, the ability of databases to quickly sort on any attribute allow the fast retrieval of similar items, e.g. those of similar size, or those created at a similar time. This can also be applied to cluster similar attributes of items in a search result, so that the user can refine the search by selecting only certain attributes. It is clear that many of the most interesting queries and clusters of data would be based on the user's activities, so usage data should be logged.

Another clear requirement for a lifetime store is support for many visualizations. Already, PC users expect to view their directories as lists, icons, or thumbnails. Just as scientists with large data sets value visualizations of their data to glean insight, users will value visualizations to gain insight into their own large data set: their lifetime store. Many tables, charts, and graphs are potentially enlightening. Furthermore, the visualization must become a UI – the user will want to click on a row of a table or a peak in a graph and see the data behind it.

It merits mention that search, while important, is not the only function of a lifetime store. Most items are likely to be forgotten so they will certainly not be searched for. These forgotten items find usefulness only when browsing and mining is supported. Mining may involve complex correlations, or it may be something as simple as random display of a photo that has not been viewed for a long time.

To summarize, our requirements include: full text search, annotation (voice, text, and handwriting), typed links, transclusion, collections, powerful queries (potentially saved), fast sorting/clustering, many visualizations, browsing, and mining. In the following sections we will explain the data model needed for such requirements and describe our implementation to date.

Data Model

"The Web isn't hypertext, it's DECORATED DIRECTORIES" – T. Nelson

Any storage system has a Logical Data Structure (LDS), consisting of various entities and their relationships [4]. An LDS is purely logical, apart from any implementation, such as in a database schema (the database schema may look very different from the LDS for efficiency and other reasons). A relationship between two entities in an LDS implies two links; one in each direction. For example the link from photograph to person may be called "photograph of person" while the same relationship from person to photograph is a link called "person in photograph".

The MyLifeBits LDS is implemented in a SQL Server database schema. It is important to understand that there is no "right" LDS for something as universal MyLifeBits; that would be to say that there is one correct LDS for all possible human knowledge. To the contrary, the LDS for MyLifeBits should reflect the worldview and needs of the particular user.[1]

That said, we do not believe that users will do much in the way of creating or altering their LDS, or the practical expression of it – the database schema. In practice, schemas are defined by applications and/or standards. We can envision power users making small extensions to a schema (perhaps adding a column to a database table). Extreme power users may do a little creation, but this will be an exception. When users do modify the schema, there is real potential for simply making things worse. E.g., they might create a new link type "about" and use it to indicate that a document is about the photo. However, this is merely to say that the document is an annotation of the photo, and they have only succeeded in making things more confusing by creating two link types (annotation and about) for the same thing. When users define new links, there is also a danger that future readers may not understand what the author meant by the link [13]. If too many link types are created, users may become too intimidated to select any type [13].

We expect that users will end up with a schema that is at least close to reflecting their worldview by obtaining the applications that create schemas, or by downloading schemas in some standard format. At present, we support easy extensibility and modification of our LDS/schema merely as a convenience for rapid prototyping and experimentation.

We have defined some entities for MyLifeBits based on some common objects on the PC, including: collection, document, image, video, song, telephone recording, email message, contact, and event. All of our entities have an ID for their key. No other fields need be unique. In particular, a collection may contain two items with the same name, unlike a directory in the file system. Furthermore, because references are to the ID of the instance, an instance may be safely moved among collections without breaking links (in contrast to HTML hyperlinks, which are to locations in a directory structure such that moving a file breaks the link).

Relationships can be easily distinguished from each other when different entities are involved. For example, there is no mistaking a relationship between a photo and person, with the relationship between a person and an event. However, there can be more than one relationship between the same entities, and such relationships must be distinguished only by name. For example, a person may be the organizer of an event, or may be an attendee of an event. These are two different types of relationships, and each link should be given a unique name so the user can recognize the difference, e.g. "organizer of", "organized by", "attendee of" and "attended by". MyLifeBits requires every link type to have a unique name for the pair of entities involved.

While there is no "correct" exhaustive list of relationships that should be supported, we do believe there are two universal relationships that should be supported by the shell: containment and annotation. They are universal in the sense that they may apply to all entities, and are also universal in human usage.

An annotation relationship indicates that one instance is making a comment upon another. The two links defined by the relationship are "annotates" (commentary on) and "annotated by" (context of the commentary). The action of comment and context is fundamental to all ongoing human discourse, especially in the scholarly realm. Any entity that could be authored to make a statement could be used to make a comment, and hence could be the source of an annotation link. Any entity can be commented on, so any entity may be the target of an annotation link.

The containment relationship indicates that one instance is contained in another. Its links are "contains" and

[1] Bates makes some interesting comments on the "fallacy of ontology" [2]

"contained in". Containment allows the universal operation of collecting and organizing things. An instance may be contained by zero or more parent instances. The concept of containment is recursive, so cycles imply an infinite loop and are not allowed (they must form a directed acyclic graph). Consequently, "contains" links can construct trees, but are not restricted to them. For most parent entities, containment indicates that the target has been authored into the source, i.e. transclusion. On the other hand, containment may be used to simply designate sets. For this, MyLifeBits has the collection entity. Entity instances linked by a containment link to a collection entity instance define a set. To a first approximation, collections exist merely so we can name sets of objects, so a name string is their only interesting attribute.

One apparently obvious and universal relationship is "related", that is, it merely indicates that two items are related in some way. However, all relationships indicate that two items are related. Also, a relationship can be indicated by putting instances in a collection with a blank name (or one named "related items"). For more than n>2 objects, using containment in a collection to designate "related" has the advantage of requiring only n links, where a fully connected graph of "related" links would require $n^2/2$ links. In a fashion similar to "related", a little thought reveals that "about" and "regarding" are really annotations.

Table 2 – Some MyLifeBits links (name is from source to target)

Link	*Source*	*Target*
Contains	Any	Any
Annotates	Document, audio, image	Any
Author of	Person	Any
Capture of	Person, Event	Image, video, audio, phone-call
Is material for	NOT(person, event)	Event
Attendee	Contact	Event
Organizer	Person	Event

In addition to our universal relationships (containment and annotation), MyLifeBits allows new relationships to be defined by the user (or an application). The endpoints of each link in a relationship are typed. For example, a contact may have an "in-photo" link to an image. However, an image cannot have an "author" link to a calendar event. MyLifeBits does not understand any semantics of user defined relationships; it merely knows the types that can be involved in the relationship. Indeed, it probably does not have any need to understand them. Apart from containment, which implies recursion in operations like backup or sharing, the shell does not do much with relationships/links apart from showing their existence, their name, and following them from one instance to another. We have created a number of relationship/link types for our own experimentation with MyLifeBits. They are some obvious and common relationships between our entities.

Table 2 shows some of the relationships/links in MyLifeBits. Figure 2 illustrates the core of the MyLifeBits schema, as implemented in SQL Server. All entities from the LDS are called "resources", and a table ("Resource entities") indicates which specific entity type a resource is. The specific entity types, such as people and images, each have their own table (boxes on the left in Figure 2). Links have pointers to two resources (source and target), and have a type. Events for resources are tracked in the event log.

The MyLifeBits System

MyLifeBits includes tools for capture, storage, retrieval, reporting, annotation, and story creation. Figure 1 illustrates the MyLifeBits system. Annotation creation must be easy, in whatever mode strikes the user's fancy, and available at the moment the fancy strikes. We have discussed ease of annotation, and storytelling as the ultimate annotation elsewhere [7].

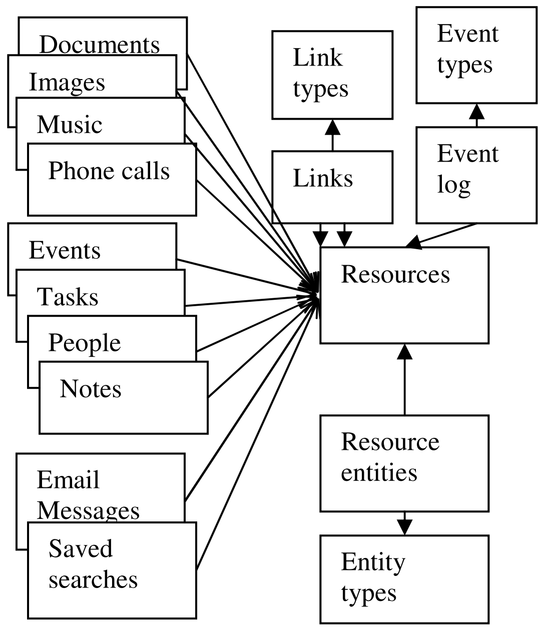


Figure 2 – Key elements of the MyLifeBits schema. Each box is a database table. Arrows indicate foreign keys.

Using a database and supporting links allows for powerful new ways of browsing, search and organization. A natural first step in organization power was to allow objects (including collections) to be contained in any number of parent collections, rather than enforcing strict hierarchy. This allows the user to file an object in as many ways as possible. While such a containment structure is indeed powerful and important, having hyperlinks causes one to re-think many of the uses of collections. For instance, a collection for a date range is much better represented by a saved query for objects matching the date range. A collection to hold objects related to an event is better represented by links from the object to a calendar event object. Similarly, a collection of objects related to a person is better represented by links from the objects to the person object. This is not because links are better than collections. After all, to be in a collection means to have a containment link to the collection. It is the more specific entity as the link target that is desirable. For example, a person object with name "Joe Smith" is a better representation for a person than a collection with the name "Joe Smith". Collections are for generic containment, and wherever possible a more specific object should be used.

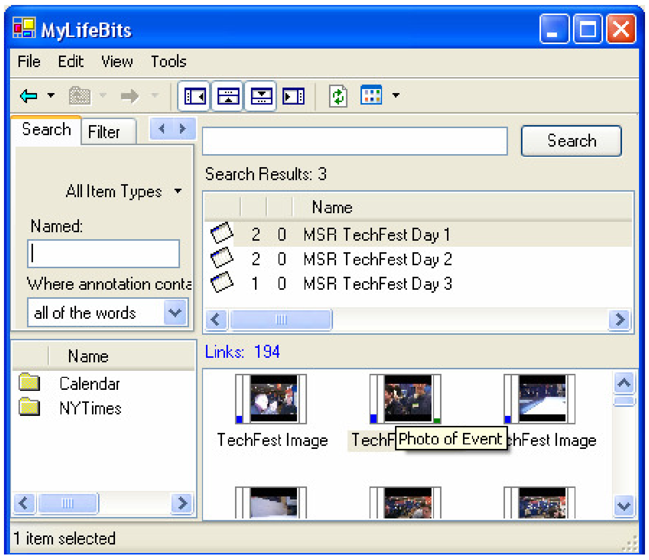


Figure 3 – Links pane (bottom right) shows instances linked to the selected item from the search results (top right). In thumbnail view, the link type is indicated by a popup on mouse hover. The bottom left pane shows "parents", i.e. instances that have a contain link to the selected item.

One advantage of strong typing is that it prevents some misfiling. For example, one could accidentally file a photo in a collection called "phone calls with Sam", but it is impossible to create a "call from" link between a photo and the "Sam" person entity, because the "call from" link is typed to be from a person to a phone call only. One could allow collections to be user-typed (i.e. to designate a collection can only contain certain items) and gain the same protection from misfiling. While this provides power when used appropriately, it can be used inappropriately to use a collection where a more specific object would be better. As in many instances, power for the educated user is a danger for the naïve user. Presently, we are building in as much expressive power as we can, even if "bad things" can be done with the system. We will learn from experience what features should be hidden or removed in the UI.

The MyLifeBits UI shows all instances linked to the currently selected instance (Figure 3). In detail view, the link type is a column. In thumbnail and timeline view, the link type is displayed as a pop-up when the mouse hovers over the item. When the selected instance is contained in other instances, they are shown in a special parent window pane.

Using a database provides fast search. User searches can be saved so that they can be repeated with a single click. Additionally, the system, or third parties, can provide complex SQL queries beyond what a user could specify in any feasible form-based UI. For example, MyLifeBits includes a "commonly used files" query that considers the number of times a file has been used, *n*, with the time since last use, *t*, and the time span of its use, *T* (i.e. the time from its first use until the time of its last use). The sort order is based on the function

$$f(n,t,T) = (t/\alpha T)^{\beta} + 1/\gamma n$$

where α, β, and γ are constants. By appropriately setting the constants, one can capture the notion that a document that has been opened over the course of a year, and then not accessed for a week, is likely to be used again. However, another document that has only been opened over the course of two days, and then not accessed for a week, is not as likely to be used again.

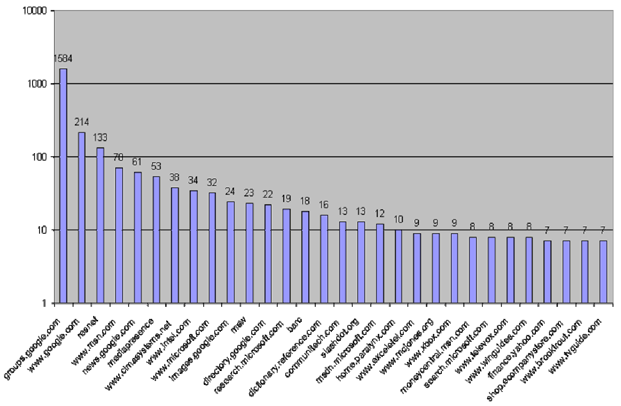


Figure 4 – MyLifeBits reports of most visited web sites

With a database, one can also create reports to understand what is in your personal store. Figure 4 shows a MyLifeBits report of the most commonly visited web sites. Figure 5 shows a report of email containing the text "funding" plotted versus time – something a user may want to see to get an idea of when busy times related to funding are happening. Excel is used to generate the graphics in the reports. Six standard reports are included in the current version of MyLifeBits, and, of course, any number could be added.

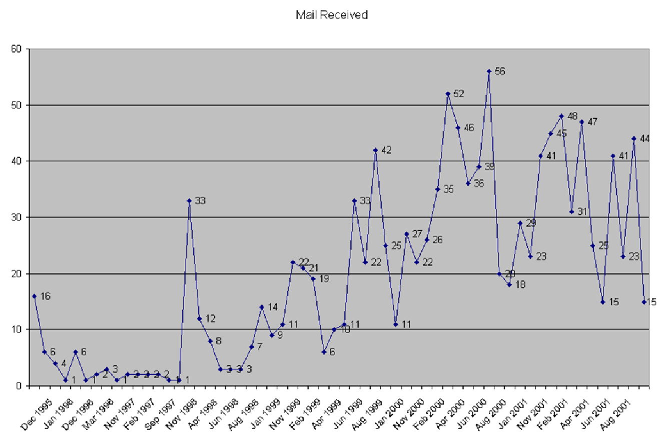
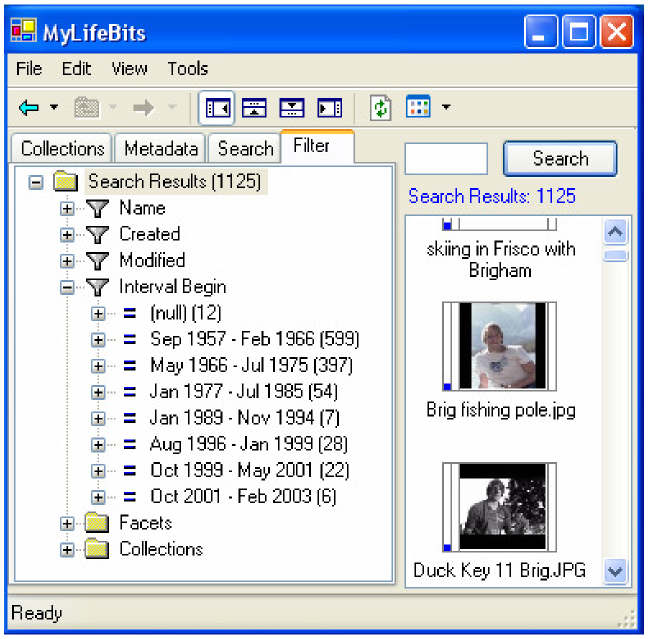


Figure 5 – MyLifeBits report: Mail received containing the text "funding" plotted versus time.

The indices in the database can be exploited to support pivoting, especially pivoting by time. That is, given some object, you can ask the system to pivot by time and show everything with a timestamp close to that of the object. The following example illustrates how hyperlinks and pivoting can be used to find objects that could not be found otherwise. Suppose that Bill has a phone call with his realtor to discuss pricing his home, and his realtor tells him the URL to open on the World Wide Web to view a comparable property. Months later, Bill remembers this property and wants to look at the page again. He knows it has been saved by the MyLifeBits browser tool. However, he cannot remember any text from the page to search for, nor does he remember when the call happened very accurately, just that it was in the fall. What Bill can do is look up his realtor from his contacts. Selecting the contact for his realtor displays a window will all items linked to the realtor. One of these items is a phone call recording with the link "caller" to the realtor's contact (this link was created at the time of the call based on the caller-id). Bill could listen to the recorded call to hear the URL, but that would be tedious. Instead, Bill right-clicks on the call and selects "pivot-by-time". This takes Bill to a time-sorted list of everything in his database, scrolled to the phone call recording. As the web page was visited soon after the call began, Bill can easily scroll a few items down and find it.



Search Results (1125)
- Name
- Created
- Modified
- Interval Begin
  - (null) (12)
  - Sep 1957 - Feb 1966 (599)
  - May 1966 - Jul 1975 (397)
  - Jan 1977 - Jul 1985 (54)
  - Jan 1989 - Nov 1994 (7)
  - Aug 1996 - Jan 1999 (28)
  - Oct 1999 - May 2001 (22)
  - Oct 2001 - Feb 2003 (6)
- Facets
- Collections

Search Results: 1125

skiing in Frisco with Brigham

Brig fishing pole.jpg

Duck Key 11 Brig.JPG

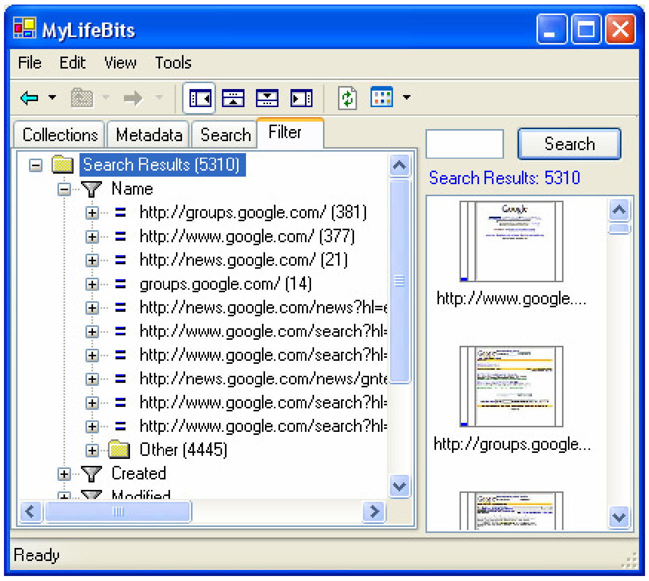
Ready

Figure 6 – Date clustering of search results using largest gap and k-means.

Use of a database also makes narrowing of search results very convenient and flexible. The MyLifeBits UI allows the user to filter results to only certain entities. For example, a search that returns photos, documents and videos, could be narrowed to only show the videos. We also perform clustering on some entity attributes for the purpose of filtering. For example, date attributes are clustered using a combination of largest-gap and k-means clustering (Figure 6). The user can click on a date cluster to narrow the search results. For text attributes like email subject or web page title, MyLifeBits shows the top seven occurring strings as clusters and puts the rest in an "other" cluster (Figure 7).

These techniques for narrowing search can also be useful as a way to browse the full contents of MyLifeBits (equivalent to narrowing a search for everything).

We have implemented two programs that suggest links to the user. Our photo capture wizard suggests that photos taken when an event in the user's calendar occurred are in fact photos of the event, and should have the corresponding link created. Our telephone recording application suggests that a contact with a phone number matching the caller-ID phone number is participating in the call.



Search Results (5310)
Name
http://groups.google.com/ (381)
http://www.google.com/ (377)
http://news.google.com/ (21)
groups.google.com/ (14)
http://news.google.com/news?hl=e
http://www.google.com/search?hl=
http://www.google.com/search?hl=
http://news.google.com/news/gnte
http://www.google.com/search?hl=
http://www.google.com/search?hl=
Other (4445)
Created
Modified

Search Results: 5310
http://www.google....
http://groups.google...

Figure 7 – Top occurring names in search result.

Conclusion

"Creative thought and essentially repetitive thought are very different things. For the latter there are, and may be, powerful mechanical aids" – V. Bush

We have entered an era of virtually unlimited storage, enabling lifetime storage of most of what one sees and hears, along with many new data source such as user logs and sensor data.

Now that we are able to have a lifetime store, the challenge is to make it useful. We have implemented a number of features proposed by Bush and Nelson to make the store useful, such as transclusion, links, annotations, and full text search. In this paper, we have shown that it is essential to augment the concepts of Bush and Nelson with typed entities/links and database features. Visualizations, complex queries, and pivots enable the user to find or browse many items that would otherwise remain unused.

The current MyLifeBits infrastructure is already very powerful, but we are at work extending it to support features such as versioning. Also, it is evident that the very power and usefulness of our infrastructure makes severe misuse possible (and evenly likely) by the naïve user. It is clear that MyLifeBits must evolve into "My Personal Assistant", with the system helping to organize and understand the data. Document similarity and face recognition are examples of important future components.

ACKNOWLEDGMENTS

MyLifeBits has benefited from the insightful comments of Jim Gray, Ted Nelson, and John Carlis. Josh Blumenstock and Evan Solomon wrote much of the code for the telephone, TV, and radio tools. Curtis Wong and Steven Drucker have helped us with UI issues.

References

[1] Eytan Adar, David Karger, and Lynn Andrea Stein. Haystack: Per-User Information Environments, In Proceedings of CIKM '99 (Kansas City, MO, November 2-6, 1999), ACM Press, 413-422.

[2] Marcia J. Bates, After the Dot-Bomb: Getting Web Information Retrieval Right This Time, First Monday, volume 7, number 7 (July 2002), http://firstmonday.org/issues/issue7_7/bates/index.html

[3] Vannevar Bush. As We May Think, The Atlantic Monthly, 176(1), July 1945, 101-108.

[4] John Carlis, Mastering Data Modeling: A User-Driven Approach, Addison-Wesley, Boston, USA, 2001.

[5] Paul Dourish, Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Michael Salisbury, Douglas Terry and Jim Thornton. Extending Document Management Systems with User-Specific Active Properties, ACM TOIS, 18(2), 2000, pp. 140-170.

[6] Freeman, Eric, Gelernter., David. LifeStreams: A storage model for personal data, ACM SIGMOD Bulletin 25, 1, March 1996, pp. 80-86.

[7] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong, MyLifeBits: Fulfilling the Memex Vision, ACM Multimedia '02, December 1-6, 2002, Juan-les-Pins, France, pp. 235-238.

[8] David Huynh, David Karger, and Dennis Quan. Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF. Semantic Web Workshop, 2002.

[9] Timothy J. Mills, David Pye, David Sinclair, Kenneth R. Wood , Shoebox: A Digital Photo Management System AT&T Technical Report 2000.10

[10] Theodor Holm Nelson, Literary Machines, `Mindful Press, Sausalito, CA, 1993.

[11] Theodor Holm Nelson. Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use, ACM Computing Surveys, 31(4es), December 1999.

[12] Rekimoto, J. Time-machine Computing: A Time-centric Approach for the Information Environment, Proceedings of UIST '99 (Asheville, NC, Nov. 1999), ACM Press, 1999, pp. 45-54.

[13] Randall Trigg, A Networked-Based Approach to Text Handling for the Online Scientific Community, PhD Dissertation, U. Maryland, Tech report TR-1346, Nov. 1983