Methods and Systems for Managing Data

Abstract: Systems and methods for managing data, such as metadata or index databases. In one exemplary method, a notification that an existing file has been modified or that a new file has been created is received by an indexing software component, which then, in response to the notification, performs an indexing operation, where the notification is either not based solely on time or user input or the notification includes an identifier that identifies the file. Other methods in data processing systems and machine readable media are also described.
METHODS AND SYSTEMS FOR MANAGING DATA

[0001] This application is a continuation-in-part of co-pending U.S. Patent Application Serial No. 10/877,584, filed on June 25, 2004. This application also claims priority to co-pending U.S. Provisional Patent Application No. 60/643,087 filed on January 7, 2005, which provisional application is incorporated herein by reference in its entirety; this application claims the benefit of the provisional's filing date under 35 U.S.C. § 119(e). This present application hereby claims the benefit of these earlier filing dates under 35 U.S.C. §120.

BACKGROUND OF THE INVENTION

[0002] Modern data processing systems, such as general purpose computer systems, allow the users of such systems to create a variety of different types of data files. For example, a typical user of a data processing system may create text files with a word processing program such as Microsoft Word or may create an image file with an image processing program such as Adobe's PhotoShop. Numerous other types of files are capable of being created or modified, edited, and otherwise used by one or more users for a typical data processing system. The large number of the different types of files that can be created or modified can present a challenge to a typical user who is seeking to find a particular file which has been created.

[0003] Modern data processing systems often include a file management system which allows a user to place files in various directories or subdirectories (e.g. folders) and allows a user to give the file a name. Further, these file management systems often allow a user to find a file by searching for the file's name, or the date of creation, or the date of modification, or the type of file. An example of such a file management system is the Finder program which operates on Macintosh computers from Apple Computer, Inc. of Cupertino, California. Another example of a file management system program is the Windows Explorer
program which operates on the Windows operating system from Microsoft Corporation of Redmond, Washington. Both the Finder program and the Windows Explorer program include a find command which allows a user to search for files by various criteria including a file name or a date of creation or a date of modification or the type of file. However, this search capability searches through information which is the same for each file, regardless of the type of file. Thus, for example, the searchable data for a Microsoft Word file is the same as the searchable data for an Adobe PhotoShop file, and this data typically includes the file name, the type of file, the date of creation, the date of last modification, the size of the file and certain other parameters which may be maintained for the file by the file management system.

[0004] Certain presently existing application programs allow a user to maintain data about a particular file. This data about a particular file may be considered metadata because it is data about other data. This metadata for a particular file may include information about the author of a file, a summary of the document, and various other types of information. A program such as Microsoft Word may automatically create some of this data when a user creates a file and the user may add additional data or edit the data by selecting the "property sheet" from a menu selection in Microsoft Word. The property sheets in Microsoft Word allow a user to create metadata for a particular file or document. However, in existing systems, a user is not able to search for metadata across a variety of different applications using one search request from the user. Furthermore, existing systems can perform one search for data files, but this search does not also include searching through metadata for those files.

[0005] Existing systems perform indexing operations in response to a set, predetermined time (e.g. set by a user) or in response to an immediate user request to begin indexing operations.

SUMMARY OF THE DESCRIPTION
[0006] Methods for managing data in a data processing system and systems for managing data are described herein.

[0007] In one aspect of the inventions described herein, an exemplary method includes receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or a new file has been created on the data storage device and, in response to the notification, performing an indexing operation on the existing file or the new file. Preferably, the notification comprises an identifier that identifies the existing file (or the new file). In another embodiment, the notification is not based solely on time or a user input. The indexing operation may be delayed or stalled or have a reduced processing priority in response to detecting user activity. The notification may be entered into a queue for indexing operations and the queue may be saved in a non-volatile storage. Changes to the queue may be entered into a transaction log of notifications.

[0008] According to another aspect of the inventions described herein, an exemplary method for processing data includes determining a file is to be indexed into an indexing database or is to have its metadata added to a metadata database, adding an entry representing the file into a list for indexing operations (or metadata operations), and saving a list to a non-volatile storage. This method may further include removing the entry from the list after indexing the file (or after metadata operations) to create an updated list and saving the updated list to the non-volatile storage. This method may further include entering changes to the list in a transaction log. The notification may be from an operating system component which provides notification in response to storing data with respect to the file on a storage device, such as the non-volatile storage.

[0009] According to another aspect of the inventions, an exemplary method includes monitoring a user’s use of a data processing system and adjusting automatically, in response to the monitoring, indexing operations or metadata processing operations. The indexing operations are typically operations which
include indexing files to create entries to be added to an index database and the metadata operations typically involve the addition of metadata, for a file, into a metadata database. The indexing operations or metadata operations may be performed automatically by the data processing system in response to the notifications, which may be from an operating system component to an indexing software component or a metadata software component. Typically, fewer indexing operations are performed over a period of time as the user’s use of the data processing system increases. The adjusting automatically of the indexing operations (or metadata operations) may involve changing the processing priority (e.g. through the Unix “Nice” command) of indexing software (or metadata software) relative to other software which is being executed on the data processing system. The priority of indexing operations may be changed or the priority of input/output (I/O) operations may be changed or the priorities of both operations may be changed.

[0010] According to another aspect of the inventions described herein, an exemplary method of processing data includes determining a time when an index database was last updated, wherein the index database contains content from files stored on a storage device, and determining whether files stored on the storage device have been modified or created after the time when the index database was last updated, and updating the index database for any file which was modified or created after the time when the index database was last updated. The updating may occur automatically without user interaction. A similar method may be performed to determine whether to update a metadata database.

[0011] According to another aspect of the inventions described herein, an exemplary method for processing data includes mounting a storage device and determining, automatically in response to the mounting, whether to index one or more files on the storage device. The determining whether to index one or more files may include comparing the last time of usage or closing of an index database to an un-mounting time or last write time of the storage device and may also
include comparing a first time of usage or opening of the index database after the
un-mounting time to a mounting time or first write time of the storage device.
Typically, the index database is stored on the storage device which is being
mounted. A similar method may be performed to determine whether to update a
metadata database automatically in response to mounting a storage device.

[0012] According to another aspect of the inventions described herein, an
exemplary method includes mounting a storage device and evaluating, after the
mounting, whether to automatically index one or more files on a storage device
by evaluating whether files on the storage device have been modified or new files
have been added since an index database for the files on the storage device was
last closed or written to.

[0013] According to another aspect of the inventions described herein, an
exemplary method for processing data includes mounting a storage device and
determining, automatically in response to the mounting, whether to index one or
more files on the storage device without examining a record for each file indexed
in the index database.

[0014] The foregoing exemplary methods may also be performed to
determine whether metadata from new files or modified files need to be
automatically added (e.g. imported) into a metadata database. For example, an
exemplary method according to such an aspect of the inventions, includes
mounting a storage device and determining, automatically in response to the
mounting whether to import metadata from one or more files on the storage
device.

[0015] Another exemplary method of processing data includes determining
whether metadata from a file is to be imported or otherwise added into a metadata
database, adding an entry representing the file into a list for importing or adding
metadata from the file into the metadata database and saving the list to a non-
volatile storage. The changes to the list may be entered into a transaction log, and
entries in the list are removed from the list after adding metadata for the file into the metadata database.

[0016] According to another aspect of the inventions described herein, an exemplary method includes monitoring a user’s use of a data processing system, and adjusting automatically, in response to the monitoring, importing or adding operations which add metadata from a file into a metadata database. Typically, the metadata database contains metadata data from a plurality of different types of files such that the type of information in metadata for files of the first type differs from the type of information in metadata for files of a second type.

[0017] Other aspects of the present invention include various data processing systems which perform these methods and machine readable media which perform one or more of these various methods described herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

[0019] Figure 1 shows an exemplary embodiment of a data processing system, which may be a general purpose computer system and which may operate in any of the various methods described herein.

[0020] Figure 2 shows a general example of one exemplary method of one aspect of the invention.

[0021] Figure 3A shows an example of the content of the particular type of metadata for a particular type of file.

[0022] Figure 3B shows another example of a particular type of metadata for another particular type of file.

[0023] Figure 4 shows an example of an architecture for managing metadata according to one exemplary embodiment of the invention.
Figure 5 is a flowchart showing another exemplary method of the present invention.

Figure 6 shows an example of a storage format which utilizes a flat file format for metadata according to one exemplary embodiment of the invention.

Figures 7A-7E show a sequence of graphical user interfaces provided by one exemplary embodiment in order to allow searching of metadata and/or other data in a data processing system.

Figures 8A and 8B show two examples of formats for displaying search results according to one exemplary embodiment of the invention.

Figure 9 shows another exemplary user interface of the present invention.

Figure 10 shows another exemplary user interface of the present invention.

Figures 11A-11D show, in sequence, another exemplary user interface according to the present invention.

Figures 12A-12D show alternative embodiments of user interfaces according to the present invention.

Figures 13A and 13B show further alternative embodiments of user interfaces according to the present invention.

Figures 14A, 14B, 14C, and 14D show further alternative embodiments of user interfaces according to the present invention.

Figures 15A, 15B, 15C and 15D show another alternative embodiment of user interfaces according to the present invention.

Figures 16A and 16B show certain aspects of embodiments of user interfaces according to the present invention.

Figure 17 shows an aspect of certain embodiments of user interfaces according to the present invention.
Figures 18A and 18B show further aspects of certain embodiments of user interfaces according to the present invention.

Figures 19A, 19B, 19C, 19D, and 19E show further illustrative embodiments of user interfaces according to the present invention.

Figure 20 is a flow chart which illustrates another exemplary method of the present invention.

Figure 21 is a flow chart showing another exemplary method of the present invention.

Figures 22A, 22B, 22C, and 22D illustrate the display of a display device on which an embodiment of the method of Figure 21 is performed.

Figure 23 is a flow chart which illustrates an exemplary method according to certain aspects of the inventions described here.

Figure 24 is a flow chart which shows an exemplary method in which indexing is performed in response to notifications about files.

Figure 25 is a flow chart which shows another exemplary method according to certain aspects of the invention.

Figure 26 is a flow chart which shows an exemplary method for determining whether to update a database.

Figure 27 is a flow chart which illustrates an exemplary method for determining whether to update a database, such as an index database or a metadata database.

Figure 28 is a flow chart which illustrates an example of a method for determining whether to update a database, such as an index database or a metadata database.

DETAILLED DESCRIPTION

The subject invention will be described with reference to numerous details set forth below, and the accompanying drawings will illustrate the invention. The following description and drawings are illustrative of the
invention and are not to be construed as limiting the invention. Numerous specific details are described to provide a thorough understanding of the present invention. However, in certain instances, well known or conventional details are not described in order to not unnecessarily obscure the present invention in detail.

[0049] The present description includes material protected by copyrights, such as illustrations of graphical user interface images. The owners of the copyrights, including the assignee of the present invention, hereby reserve their rights, including copyright, in these materials. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office file or records, but otherwise reserves all copyrights whatsoever. Copyright Apple Computer, Inc. 2004.

[0050] Figure 1 shows one example of a typical computer system which may be used with the present invention. Note that while Figure 1 illustrates various components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the components as such details are not germane to the present invention. It will also be appreciated that network computers and other data processing systems which have fewer components or perhaps more components may also be used with the present invention. The computer system of Figure 1 may, for example, be a Macintosh computer from Apple Computer, Inc.

[0051] As shown in Figure 1, the computer system 101, which is a form of a data processing system, includes a bus 102 which is coupled to a microprocessor(s) 103 and a ROM (Read Only Memory) 107 and volatile RAM 105 and a non-volatile memory 106. The microprocessor 103 may be a G3 or G4 microprocessor from Motorola, Inc. or one or more G5 microprocessors from IBM. The bus 102 interconnects these various components together and also interconnects these components 103, 107, 105, and 106 to a display controller and display device 104 and to peripheral devices such as input/output (I/O) devices.
which may be mice, keyboards, modems, network interfaces, printers and other
devices which are well known in the art. Typically, the input/output devices 109
are coupled to the system through input/output controllers 108. The volatile
RAM (Random Access Memory) 105 is typically implemented as dynamic RAM
(DRAM) which requires power continually in order to refresh or maintain the
data in the memory. The mass storage 106 is typically a magnetic hard drive or a
magnetic optical drive or an optical drive or a DVD RAM or other types of
memory systems which maintain data (e.g. large amounts of data) even after
power is removed from the system. Typically, the mass storage 106 will also be a
random access memory although this is not required. While Figure 1 shows that
the mass storage 106 is a local device coupled directly to the rest of the
components in the data processing system, it will be appreciated that the present
invention may utilize a non-volatile memory which is remote from the system,
such as a network storage device which is coupled to the data processing system
through a network interface such as a modem or Ethernet interface. The bus 102
may include one or more buses connected to each other through various bridges,
controllers and/or adapters as is well known in the art. In one embodiment the
I/O controller 108 includes a USB (Universal Serial Bus) adapter for controlling
USB peripherals and an IEEE 1394 controller for IEEE 1394 compliant
peripherals.

[0052] It will be apparent from this description that aspects of the present
invention may be embodied, at least in part, in software. That is, the techniques
may be carried out in a computer system or other data processing system in
response to its processor, such as a microprocessor, executing sequences of
instructions contained in a memory, such as ROM 107, RAM 105, mass storage
106 or a remote storage device. In various embodiments, hardwired circuitry may
be used in combination with software instructions to implement the present
invention. Thus, the techniques are not limited to any specific combination of
hardware circuitry and software nor to any particular source for the instructions

-10-
executed by the data processing system. In addition, throughout this description, various functions and operations are described as being performed by or caused by software code to simplify description. However, those skilled in the art will recognize what is meant by such expressions is that the functions result from execution of the code by a processor, such as the microprocessor 103.

**Capturing and Use of Metadata Across a Variety of Application Programs**

[0053] **Figure 2** shows a generalized example of one embodiment of the present invention. In this example, captured metadata is made available to a searching facility, such as a component of the operating system which allows concurrent searching of all metadata for all applications having captured metadata (and optionally for all non-metadata of the data files). The method of **Figure 2** may begin in operation 201 in which metadata is captured from a variety of different application programs. This captured metadata is then made available in operation 203 to a searching facility, such as a file management system software for searching. This searching facility allows, in operation 205, the searching of metadata across all applications having captured metadata. The method also provides, in operation 207, a user interface of a search engine and the search results which are obtained by the search engine. There are numerous possible implementations of the method of **Figure 2**. For example, **Figure 5** shows a specific implementation of one exemplary embodiment of the method of **Figure 2**. Alternative implementations may also be used. For example, in an alternative implementation, the metadata may be provided by each application program to a central source which stores the metadata for use by searching facilities and which is managed by an operating system component, which may be, for example, the metadata processing software. The user interface provided in operation 207 may take a variety of different formats, including some of the examples described below as well as user interfaces which are conventional, prior art user interfaces. The metadata may be stored in a database which may be any of a variety of
formats including a B tree format or, as described below, in a flat file format according to one embodiment of the invention.

[0054] The method of Figure 2 may be implemented for programs which do not store or provide metadata. In this circumstance, a portion of the operating system provides for the capture of the metadata from the variety of different programs even though the programs have not been designed to provide or capture metadata. For those programs which do allow a user to create metadata for a particular document, certain embodiments of the present invention may allow the exporting back of captured metadata back into data files for applications which maintain metadata about their data files.

[0055] The method of Figure 2 allows information about a variety of different files created by a variety of different application programs to be accessible by a system wide searching facility, which is similar to the way in which prior art versions of the Finder or Windows Explorer can search for file names, dates of creation, etc. across a variety of different application programs. Thus, the metadata for a variety of different files created by a variety of different application programs can be accessed through an extension of an operating system, and an example of such an extension is shown in Figure 4 as a metadata processing software which interacts with other components of the system and will be described further below.

[0056] Figures 3A and 3B show two different metadata formats for two different types of data files. Note that there may be no overlap in any of the fields; in other words, no field in one type of metadata is the same as any field in the other type of metadata. Metadata format 301 may be used for an image file such as a JPEG image file. This metadata may include information such as the image's width, the image's height, the image's color space, the number of bits per pixel, the ISO setting, the flash setting, the F/stop of the camera, the brand name of the camera which took the image, user-added keywords and other fields, such as a field which uniquely identifies the particular file, which identification is

-12-
persistent through modifications of the file. Metadata format 331 shown in
**Figure 3B** may be used for a music file such as an MP3 music file. The data in
this metadata format may include an identification of the artist, the genre of the
music, the name of the album, song names in the album or the song name of the
particular file, song play times or the song play time of a particular song and other
fields, such as a persistent file ID number which identifies the particular MP3 file
from which the metadata was captured. Other types of fields may also be used.
The following chart shows examples of the various fields which may be used in
metadata for various types of files.
<table>
<thead>
<tr>
<th>Item name</th>
<th>Parent in hierarchy</th>
<th>Attribute name</th>
<th>Description/Notes</th>
<th>CFType</th>
<th>Multi-value</th>
<th>Localized</th>
<th>User settable</th>
<th>Gettable</th>
<th>Copied with copy</th>
<th>App viewable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td></td>
<td></td>
<td>Who created or contributed to the contents of this item</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Address Book</td>
</tr>
<tr>
<td>Comment</td>
<td></td>
<td></td>
<td>A free form text comment</td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ContentType</td>
<td></td>
<td></td>
<td>This is the type that is determined by UTI</td>
<td>CFString</td>
<td>No</td>
<td>?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ContentTypes</td>
<td></td>
<td></td>
<td>This is the inheritance of the UTI system</td>
<td>CFString</td>
<td>Yes</td>
<td>?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CreatedDate</td>
<td></td>
<td></td>
<td>When was this item created</td>
<td>CFDate</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DisplayName</td>
<td></td>
<td></td>
<td>The name of the item as the user would like to read it. Very well may be the file name, but it may also be the subject of an e-mail message or the full name of a person, for example.</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Finder (or Launch Services)</td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
<td></td>
<td>This is a list words set by the user to identify arbitrary sets of organization. The scope is determined by the user and can be flexibly used for any kind of organization. For example, Family, Hawaii, Project X, etc.</td>
<td>CFString</td>
<td>Yes</td>
<td>System-provided keywords (if any)</td>
<td>Yes</td>
<td>Yes</td>
<td>Ask</td>
<td></td>
</tr>
<tr>
<td><strong>Contact Keywords</strong></td>
<td>A list of contacts that are associated with this document, beyond what is captured as Author. This may be a person who's in the picture or a document about a person or contact (performance review, contract)</td>
<td><strong>CFString</strong></td>
<td><strong>Yes</strong></td>
<td><strong>No</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Ask</strong></td>
<td><strong>Address Book</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ModifiedDate</strong></td>
<td>When this item was last modified</td>
<td><strong>CFDate</strong></td>
<td><strong>No</strong></td>
<td><strong>No</strong></td>
<td><strong>No</strong></td>
<td><strong>Yes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settlable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------------</td>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td>A relative rating (0 to 5 value) on how important a particular item is to you, whether it's a person, file or message</td>
<td>CFNumber</td>
<td>No</td>
<td>n/a</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>RelatedTos</td>
<td></td>
<td></td>
<td>A list of other items that are arbitrarily grouped together.</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TextContent</td>
<td></td>
<td></td>
<td>An indexed version of any content text</td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UsedDates</td>
<td></td>
<td></td>
<td>Which days was the document opened/viewed/played</td>
<td>CFDates</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copyright</td>
<td></td>
<td></td>
<td>Specifies the owner of this content, i.e. Copyright Apple Computer, Inc.</td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreatorApp</td>
<td></td>
<td></td>
<td>Keeps track of the application that was used to create this document (if it's known).</td>
<td>CFString</td>
<td>No</td>
<td>?</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Languages</td>
<td></td>
<td></td>
<td>The languages that this document is composed in (for either text or audio-based media)</td>
<td>CFString</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content/Data</td>
<td>Item</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------</td>
<td>------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>ParentalControl</td>
<td>A field that is used to determine whether this is kid-friendly content or not</td>
<td>CFString</td>
<td>No</td>
<td>?</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publishers</td>
<td>The name or a person or organization that published this content.</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Address Book</td>
<td></td>
</tr>
<tr>
<td>PublishedDate</td>
<td>The original date that this content was published (if it was), independent of created date.</td>
<td>CFDDate</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reviewers</td>
<td>A list of contacts who have reviewed the contents of this file. This would have to be set explicitly by an application.</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Address Book</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item name</th>
<th>Parent in hierarchy</th>
<th>Attribute name</th>
<th>Description/Notes</th>
<th>CFType</th>
<th>Multi-value</th>
<th>Localized</th>
<th>User settable</th>
<th>Gettable</th>
<th>Copied with copy</th>
<th>App viewable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReviewStatus</td>
<td></td>
<td></td>
<td>Free form text that used to specify where the document is in any arbitrary review process</td>
<td>CFString</td>
<td>No</td>
<td>?</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TimeEdited</td>
<td></td>
<td></td>
<td>Total time spent editing document</td>
<td>CFDate</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WhereTos</td>
<td></td>
<td></td>
<td>Where did this go to, eg. CD, printed, backedup</td>
<td>CFString</td>
<td>Yes</td>
<td>System-provided words only (if any)</td>
<td>?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>Data</td>
<td>WhereFroms</td>
<td>CFString</td>
<td>Yes</td>
<td>System-provided words only (if any)</td>
<td>?</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------</td>
<td>---------------------------------</td>
<td>---------------</td>
<td>-----</td>
<td>--------------------------------------</td>
<td>---</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where did this come from, e.g. camera, email, web download, CD</td>
<td>CFString</td>
<td>Yes</td>
<td>System-provided words only (if any)</td>
<td>?</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BitsPerSample</td>
<td>What is the bit depth of the image (8-bit, 16-bit, etc.)</td>
<td>CFNumber</td>
<td>No</td>
<td>Yes</td>
<td>ColorSync Utility?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ColorSpace</td>
<td>What color space model is this document following</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td>ColorSync Utility?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageHeight</td>
<td>The height of the image in pixels</td>
<td>CFNumber</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageWidth</td>
<td>The width of the image in pixels</td>
<td>CFNumber</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProfileName</td>
<td>The name of the color profile used with for image</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td>ColorSync Utility?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResolutionWidth</td>
<td>Resolution width of this image (i.e. dpi from a scanner)</td>
<td>CFNumber</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResolutionHeight</td>
<td>Resolution height of this image (i.e. dpi from a scanner)</td>
<td>CFNumber</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerNames</td>
<td>For image formats that contain &quot;named&quot; layers (e.g. Photoshop files)</td>
<td>CFString</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aperture</td>
<td>The f-stop rating of the camera when the image was taken</td>
<td>CFNumber</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>CameraMake</td>
<td></td>
<td></td>
<td>The make of the camera that was used to acquire this image (e.g. Nikon)</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CameraModel</td>
<td></td>
<td></td>
<td>The model of the camera used to acquire this image (Coolpix 5700)</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DateTimeOriginal</td>
<td></td>
<td>Date/time the picture was taken</td>
<td>CFDate</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExposureMode</td>
<td></td>
<td>Mode that was used for the exposure</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExposureTime</td>
<td></td>
<td>Time that the lens was exposed while taking the picture</td>
<td>CFDate</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash</td>
<td></td>
<td>This attribute is overloaded with information about red-eye reduction. This is not a binary value</td>
<td>CFNumbe...</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td></td>
<td>Raw value received from GPS device associated with photo acquisition. It hasn't necessarily been translated to a user-understandable location.</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISOSpeed</td>
<td></td>
<td>The ISO speed the camera was set to when the image was acquired</td>
<td>CFNumbe...</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orientation</td>
<td>The orientation of the camera when the image was acquired</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------</td>
<td>----------</td>
<td>----</td>
<td>---</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WhiteBalance</td>
<td>The white balance setting of the camera when the picture was taken</td>
<td>CFNumbe</td>
<td>No</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIFversion</td>
<td>The version of EXIF that was used to generate the metadata for the image</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AcquisitionSources</td>
<td>The name or type of device that used to acquire the media</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multivalued</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Time-based</td>
<td>Codecs</td>
<td>The codecs used to encode/decode the media</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td>DeliveryType</td>
<td>FastStart or RTSP</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration</td>
<td>The length of time that the media lasts</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Streamable</td>
<td>Whether the content is prepared for purposes of streaming</td>
<td>CFBoolean</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TotalBitRate</td>
<td>The total bit rate (audio &amp; video combined) of the media.</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AudioBitRate</td>
<td>The audio bit rate of the media</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AspectRatio</td>
<td>The aspect ratio of the video of the media</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ColorSpace</td>
<td>The color space model used for the video aspect of the media</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FrameHeight</td>
<td>The frame height in pixels of the video in the media</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FrameWidth</td>
<td>The frame width in pixels of the video in the media</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProfileName</td>
<td>The name of the color profile used on the video portion of the media</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Data</td>
<td>VideoBitRate</td>
<td>The bit rate of the video aspect of the media</td>
<td>CFNumbe</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------------------------------------------</td>
<td>----------</td>
<td>------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subject</td>
<td></td>
<td>The subject of the text. This could be metadata that's supplied with the text or something automatically generated with technologies like VTWIN</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PageCount</td>
<td></td>
<td>The number of printable pages of the document</td>
<td>CFNumbe</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LineCount</td>
<td></td>
<td>The number of lines in the document</td>
<td>CFNumbe</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------------</td>
<td>----------</td>
<td>--------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>WordCount</td>
<td></td>
<td>The number of words in the document</td>
<td>CFNumbe r</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
<td>The URL that will get you to this document (or at least did at one time). Relevant for saved HTML documents, bookmarks, RSS feeds, etc.</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pageTitle</td>
<td></td>
<td>The title of a web page. Relevant to HTML or bookmark documents</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>google hierarchy</td>
<td></td>
<td>Structure of where this page can be found in the Google hierarchy. Relevant to HTML or bookmark documents</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compound document</td>
<td>Data</td>
<td>&lt;Abstract&gt;</td>
<td>There are no specific attributes assigned to this item. This is to catch all app-specific file formats that fall within Data, but don't fit into any of the other types. Typically these documents have multiple types of media embedded within them. (e.g. P</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>PDF Component</td>
<td>Description</td>
<td>Type</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------</td>
<td>--------------</td>
<td>----</td>
<td>-----</td>
<td>---</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NumberOfPages</td>
<td>The number of printable pages in the document</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PageSize</td>
<td>The size of the page stored as points</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDFTitle</td>
<td>PDF-specific title metadata for the document</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDFAuthor</td>
<td>PDF-specific author metadata for the document</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDFSubject</td>
<td>PDF-specific subject metadata for the document</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDFKeywords</td>
<td>PDF-specific keywords metadata for the document</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDFCreated</td>
<td>PDF-specific created metadata for the document</td>
<td>CFDatum</td>
<td>No</td>
<td></td>
<td></td>
<td>?</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDFModified</td>
<td>PDF-specific modified metadata for the document</td>
<td>CFDatum</td>
<td>No</td>
<td></td>
<td></td>
<td>?</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDFVersion</td>
<td>PDF-specific version metadata for the document</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>?</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SecurityMethod</td>
<td>Method by which this document is kept secure</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>?</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Presentation</td>
<td>Compound document</td>
<td>SlideTitles</td>
<td>A collection of the titles on slides</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Keynote)</td>
<td></td>
<td>SlideCount</td>
<td>The number of slides</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SpeakerNotesCon</td>
<td>The content of all the speaker notes from all of the slides together</td>
<td>CFString</td>
<td>?</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td>Categories</td>
<td>The kind of application this is: productivity, games, utility, graphics, etc. A set list that</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recipients</td>
<td>Maps to To and Cc: addresses in a mail message.</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td>Address Book</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Priority</td>
<td>The priority of the message as set by the sender</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Item</td>
<td>Description</td>
<td>Type</td>
<td>M</td>
<td>I</td>
<td>P</td>
<td>S</td>
<td>Yes</td>
<td>Yes</td>
<td>Address Book</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>AttachmentNames</td>
<td>The list of filenames that represent attachments in a particular message (should be actionable within the Finder)</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Authors</td>
<td>maps to From address in mail message</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comment</td>
<td>Not applicable to Mail right now (should we consider?)</td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ContentType</td>
<td></td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ContentTypes</td>
<td></td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CreatedDate</td>
<td>When was this message was sent or received</td>
<td>CFDate</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DisplayName</td>
<td>Subject of the message</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td>---------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
<td></td>
<td>There will be a way to set keywords within Mail</td>
<td>CFString</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Ask</td>
</tr>
<tr>
<td>Contact Keywords</td>
<td></td>
<td></td>
<td>Could be where recipients are held</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Address Book</td>
</tr>
<tr>
<td>ModifiedDate</td>
<td></td>
<td></td>
<td>Not applicable</td>
<td>CFDates</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td>A relative rating (0 to 5 stars) on how important a particular message is to you (separate from a message's Priority)</td>
<td>CFNNumber</td>
<td>No</td>
<td>n/a</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RelatedTos</td>
<td></td>
<td></td>
<td>Potentially threaded messages could be put into this category</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextContent</td>
<td></td>
<td></td>
<td>An indexed version of the mail message</td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UsedDates</td>
<td></td>
<td></td>
<td>The day/time in which the mail message was viewed/read</td>
<td>CFDates</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td></td>
<td></td>
<td>The company that this contact is an employee of</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Address Book</td>
<td></td>
</tr>
<tr>
<td>E-mails</td>
<td></td>
<td></td>
<td>A list of e-mail addresses that this contact has</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Mail</td>
<td></td>
</tr>
<tr>
<td>IMs</td>
<td></td>
<td></td>
<td>A list of instant message handles this contact has</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td>iChat</td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td>Item</td>
<td>Description</td>
<td>Type</td>
<td>Is Indexed</td>
<td>Comment</td>
<td>Content Type</td>
<td>ContentType</td>
<td>Content Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
<td>------------</td>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Phones</strong></td>
<td>A list of phone numbers that relate to this contact</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Addresses</strong></td>
<td>A list of physical addresses that relate to this person</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Authors</strong></td>
<td>the name of the owner of the Address Book (current user name)</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Comment</strong></td>
<td></td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ContentType</strong></td>
<td></td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ContentTypes</strong></td>
<td></td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settable</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-----------</td>
<td>--------------</td>
<td>----------</td>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CreatedDate</td>
<td>date the user entered this into his AddressBook (either through import or direct entry)</td>
<td>CFCDate</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayName</td>
<td>Composite name of contact (First Name, Last Name)</td>
<td>CFString</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Keywords</td>
<td>There will be a way to set keywords within Address Book</td>
<td>CFString</td>
<td>Yes</td>
<td>System-provided keywords (if any)</td>
<td>Yes</td>
<td>Yes</td>
<td>Ask</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContactKeywords</td>
<td></td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Ask</td>
<td>Address Book</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ModifiedDate</td>
<td>Last time this contact entry was modified</td>
<td>CFCDate</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rating</td>
<td>A relative rating (0 to 5 stars) on how important a particular contact is to you (separate from a message's Priority)</td>
<td>CFNumber</td>
<td>No</td>
<td>n/a</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RelatedTos</td>
<td>(potentially could be used to associate people from the same company or family)</td>
<td>CFString</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextContent</td>
<td>An indexed version of the Notes section</td>
<td>CFString</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UsedDates</td>
<td>The day/time in which the contact entry was viewed in Address Book</td>
<td>CFCDate</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meeting (TBD)</td>
<td>Item</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
<td>Address Book</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>text describing the event</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EventTimes</td>
<td>time/date the event starts</td>
<td>CFDatetime</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>The length of time that the meeting lasts</td>
<td>CFNumber</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invitees</td>
<td>The list of people who are invited to the meeting</td>
<td>CFString</td>
<td>Yes</td>
<td></td>
<td>Address Book</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item name</td>
<td>Parent in hierarchy</td>
<td>Attribute name</td>
<td>Description/Notes</td>
<td>CFType</td>
<td>Multi-value</td>
<td>Localized</td>
<td>User settabl e</td>
<td>Gettable</td>
<td>Copied with copy</td>
<td>App viewable</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>--------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------------</td>
<td>---------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Location</td>
<td></td>
<td></td>
<td>The name of the location where the meeting is taking place</td>
<td>CFString</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
One particular field which may be useful in the various metadata formats would be a field which includes an identifier of a plug in or other software element which may be used to capture metadata from a data file and/or export metadata back to the creator application.

Various different software architectures may be used to implement the functions and operations described herein. The following discussion provides one example of such an architecture, but it will be understood that alternative architectures may also be employed to achieve the same or similar results. The software architecture shown in Figure 4 is an example which is based upon the Macintosh operating system. The architecture 400 includes a metadata processing software 401 and an operating system (OS) kernel 403 which is operatively coupled to the metadata processing software 401 for a notification mechanism which is described below. The metadata processing software 401 is also coupled to other software programs such as a file system graphical user interface software 405 (which may be the Finder), an email software 407, and other applications 409. These applications are coupled to the metadata processing software 401 through client application program interface 411 which provide a method for transferring data and commands between the metadata processing software 401 and the software 405, 407, and 409. These commands and data may include search parameters specified by a user as well as commands to perform searches from the user, which parameters and commands are passed to the metadata processing software 401 through the interface 411. The metadata processing software 401 is also coupled to a collection of importers 413 which extract data from various applications. In particular, in one exemplary embodiment, a text importer is used to extract text and other information from word processing or text processing files created by word processing programs such as Microsoft Word, etc. This extracted information is the metadata for a particular file. Other types of importers extract metadata from other types of files, such as image files or music files. In this particular embodiment, a
particular importer is selected based upon the type of file which has been created and modified by an application program. For example, if the data file was created by PhotoShop, then an image importer for PhotoShop may be used to input the metadata from a PhotoShop data file into the metadata database 415 through the metadata processing software 401. On the other hand, if the data file is a word processing document, then an importer designed to extract metadata from a word processing document is called upon to extract the metadata from the word processing data file and place it into the metadata database 415 through the metadata processing software 401. Typically, a plurality of different importers may be required in order to handle the plurality of different application programs which are used in a typical computer system. The importers 413 may optionally include a plurality of exporters which are capable of exporting the extracted metadata for particular types of data files back to property sheets or other data components maintained by certain application programs. For example, certain application programs may maintain some metadata for each data file created by the program, but this metadata is only a subset of the metadata extracted by an importer from this type of data file. In this instance, the exporter may export back additional metadata or may simply insert metadata into blank fields of metadata maintained by the application program.

The software architecture 400 also includes a file system directory 417 for the metadata. This file system directory keeps track of the relationship between the data files and their metadata and keeps track of the location of the metadata object (e.g. a metadata file which corresponds to the data file from which it was extracted) created by each importer. In one exemplary embodiment, the metadata database is maintained as a flat file format as described below, and the file system directory 417 maintains this flat file format. One advantage of a flat file format is that the data is laid out on a storage device as a string of data without references between fields from one metadata file (corresponding to a particular data file) to another metadata file (corresponding to another data file).
This arrangement of data will often result in faster retrieval of information from the metadata database 415.

[0060] The software architecture 400 of Figure 4 also includes find by content software 419 which is operatively coupled to a database 421 which includes an index of files. The index of files represents at least a subset of the data files in a storage device and may include all of the data files in a particular storage device (or several storage devices), such as the main hard drive of a computer system. The index of files may be a conventional indexed representation of the content of each document. The find by content software 419 searches for words in that content by searching through the database 421 to see if a particular word exists in any of the data files which have been indexed. The find by content software functionality is available through the metadata processing software 401 which provides the advantage to the user that the user can search concurrently both the index of files in the database 421 (for the content within a file) as well as the metadata for the various data files being searched.

The software architecture shown in Figure 4 may be used to perform the method shown in Figure 5 or alternative architectures may be used to perform the method of Figure 5.

[0061] The method of Figure 5 may begin in operation 501 in which a notification of a change for a file is received. This notification may come from the OS kernel 403 which notifies the metadata processing software 401 that a file has been changed. This notification may come from sniffer software elements which detect new or modified files and deletion of files. This change may be the creation of a new file or the modification of an existing file or the deletion of an existing file. The deletion of an existing file causes a special case of the processing method of Figure 5 and is not shown in Figure 5. In the case of a deletion, the metadata processing software 401, through the use of the file system directory 417, deletes the metadata file in the metadata database 415 which corresponds to the deleted file. The other types of operations, such as the creation
of a new file or the modification of an existing file, causes the processing to proceed from operation 501 to operation 503 in which the type of file which is the subject of the notification is determined. The file may be an Acrobat PDF file or an RTF word processing file or a JPEG image file, etc. In any case, the type of the file is determined in operation 503. This may be performed by receiving from the OS kernel 403 the type of file along with the notification or the metadata processing software 401 may request an identification of the type of file from the file system graphical user interface software 405 or similar software which maintains information about the data file, such as the creator application or parent application of the data file. It will be understood that in one exemplary embodiment, the file system graphical user interface software 405 is the Finder program which operates on the Macintosh operating system. In alternative embodiments, the file system graphical user interface system may be Windows Explorer which operates on Microsoft's Windows operating system. After the type of file has been determined in operation 503, the appropriate capture software (e.g. one of the importers 413) is activated for the determined file type. The importers may be a plug-in for the particular application which created the type of file about which notification is received in operation 501. Once activated, the importer or capture software imports the appropriate metadata (for the particular file type) into the metadata database, such as metadata database 415 as shown in operation 507. Then in operation 509, the metadata is stored in the database. In one exemplary embodiment, it may be stored in a flat file format. Then in operation 511, the metadata processing software 401 receives search parameter inputs and performs a search of the metadata database (and optionally also causes a search of non-metadata sources such as the index of files 421) and causes the results of the search to be displayed in a user interface. This may be performed by exchanging information between one of the applications, such as the software 405 or the software 407 or the other applications 409 and the metadata processing software 401 through the interface 411. For example, the
file system software 405 may present a graphical user interface, allowing a user to input search parameters and allowing the user to cause a search to be performed. This information is conveyed through the interface 411 to the metadata processing software 401 which causes a search through the metadata database 415 and also may cause a search through the database 421 of the indexed files in order to search for content within each data file which has been indexed. The results from these searches are provided by the metadata processing software 401 to the requesting application which, in the example given here, was the software 405, but it will be appreciated that other components of software, such as the email software 407, may be used to receive the search inputs and to provide a display of the search results. Various examples of the user interface for inputting search requests and for displaying search results are described herein and shown in the accompanying drawings.

[0062] It will be appreciated that the notification, if done through the OS kernel, is a global, system wide notification process such that changes to any file will cause a notification to be sent to the metadata processing software. It will also be appreciated that in alternative embodiments, each application program may itself generate the necessary metadata and provide the metadata directly to a metadata database without the requirement of a notification from an operating system kernel or from the intervention of importers, such as the importers 413. Alternatively, rather than using OS kernel notifications, an embodiment may use software calls from each application to a metadata processing software which receives these calls and then imports the metadata from each file in response to the call.

[0063] As noted above, the metadata database 415 may be stored in a flat file format in order to improve the speed of retrieval of information in most circumstances. The flat file format may be considered to be a non-B tree, non-hash tree format in which data is not attempted to be organized but is rather stored as a stream of data. Each metadata object or metadata file will itself
contain fields, such as the fields shown in the examples of Figures 3A and 3B. However, there will typically be no relationship or reference or pointer from one field in one metadata file to the corresponding field (or another field) in the next metadata file or in another metadata file of the same file type. Figure 6 shows an example of the layout in a flat file format of metadata. The format 601 includes a plurality of metadata files for a corresponding plurality of data files. As shown in Figure 6, metadata file 603 is metadata from file 1 of application A and may be referred to as metadata file A1. Similarly, metadata file 605 is metadata from file 1 of application B and may be referred to as metadata file B1. Each of these metadata files typically would include fields which are not linked to other fields and which do not contain references or pointers to other fields in other metadata files. It can be seen from Figure 6 that the metadata database of Figure 6 includes metadata files from a plurality of different applications (applications A, B, and C) and different files created by each of those applications. Metadata files 607, 609, 611, and 617 are additional metadata files created by applications A, B, and C as shown in Figure 6.

A flexible query language may be used to search the metadata database in the same way that such query languages are used to search other databases. The data within each metadata file may be packed or even compressed if desirable. As noted above, each metadata file, in certain embodiments, will include a persistent identifier which uniquely identifies its corresponding data file. This identifier remains the same even if the name of the file is changed or the file is modified. This allows for the persistent association between the particular data file and its metadata.

User Interface Aspects

Various different examples of user interfaces for inputting search parameters and for displaying search results are provided herein. It will be understood that some features from certain embodiments may be mixed with
other embodiments such that hybrid embodiments may result from these combinations. It will be appreciated that certain features may be removed from each of these embodiments and still provide adequate functionality in many instances.

[0066] **Figure 7A** shows a graphical user interface which is a window which may be displayed on a display device which is coupled to a data processing system such as a computer system. The window 701 includes a side bar having two regions 703A, which is a user-configurable region, and 703B, which is a region which is specified by the data processing system. Further details in connection with these side bar regions may be found in co-pending U.S. Patent Application Serial No. __________ filed June 22, 2004, and entitled "Methods and Apparatuses for Operating a Data Processing System," by inventors Donald Lindsay and Bas Ording, attorney docket number 04860.P3306. The window 701 also includes a display region 705 which in this case displays the results of searches requested by the user. The window 701 also includes a search parameter menu bar 707 which includes configurable pull down menus 713, 715, and 717. The window 701 also includes a text entry region 709 which allows a user to enter text as part of the search query or search parameters. The button 711 may be a start search button which a user activates in order to start a search based upon the selected search parameters. Alternatively, the system may perform a search as soon as it receives any search parameter inputs or search queries from the user rather than waiting for a command to begin the search. The window 701 also includes a title bar 729 which may be used in conjunction with a cursor control device to move, in a conventional manner, the window around a desktop which is displayed on a display device. The window 701 also includes a close button 734, a minimize button 735, and a resize button 736 which may be used to close or minimize or resize, respectively, the window. The window 701 also includes a resizing control 731 which allows a user to modify the size of the window on a display device. The window 701 further includes a back button 732
and a forward button 733 which function in a manner which is similar to the back and forward buttons on a web browser, such as Internet Explorer or Safari. The window 701 also includes view controls which include three buttons for selecting three different types of views of the content within the display region 705. When the contents found in a search exceed the available display area of a display region 705, scroll controls, such as scroll controls 721, 722, and 723, appear within the window 701. These may be used in a conventional manner, for example, by dragging the scroll bar 721 within the scroll region 721A using conventional graphical user interface techniques.

The combination of text entry region 709 and the search parameter menu bar allow a user to specify a search query or search parameters. Each of the configurable pull down menus presents a user with a list of options to select from when the user activates the pull down menu. As shown in Figure 7A, the user has already made a selection from the configurable pull down menu 713 to specify the location of the search, which in this case specifies that the search will occur on the local disks of the computer systems. Configurable pull down menu 715 has also been used by the user to specify the kind of document which is to be searched for, which in this case is an image document as indicated by the configurable pull down menu 715 which indicates "images" as the selected configuration of this menu and hence the search parameter which it specifies.

The configurable pull down menu 717, as shown in Figure 7A, represents an add search parameter pull down menu. This add search parameter pull down menu allows the user to add additional criteria to the search query to further limit the search results. In the embodiment shown in Figure 7A, each of the search parameters is logically ANDed in a Boolean manner. Thus the current search parameter specified by the user in the state shown in Figure 7A searches all local disks for all images, and the user is in the middle of the process of selecting another search criteria by having selected the add search criteria pull down menu.
717, resulting in the display of the pull down menu 719, which has a plurality of options which may be selected by the user.

[0068] **Figure 7B** shows the window 701 after the user has caused the selection of the time option within pull down menu 719, thereby causing the display of a submenu 719A which includes a list of possible times which the user may select from. Thus it appears that the user wants to limit the search to all images on all local disks within a certain period of time which is to be specified by making a selection within the submenu 719A.

[0069] **Figure 7C** shows the window 701 on the display of a data processing system after the user has selected a particular option (in this case "past week") from the submenu 719A. If the user accepts this selection, then the display shown in **Figure 7D** results in which the configurable pull down menu 718 is displayed showing that the user has selected as part of the search criteria files that have been created or modified in the past week. It can be seen from **Figure 7D** that the user can change the particular time selected from this pull down menu 718 by selecting another time period within the pull down menu 718A shown in **Figure 7D**. Note that the configurable pull down menu 717, which represents an add search parameter menu, has now moved to the right of the configurable pull down menu 718. The user may add further search parameters by pressing or otherwise activating the configurable pull down menu 717 from the search parameter menu bar 707. If the user decides that the past week is the proper search criteria in the time category, then the user may release the pull down menu 718A from being displayed in a variety of different ways (e.g. the user may release the mouse button which was being depressed to keep the pull down menu 718A on the display). Upon releasing or otherwise dismissing the pull down menu 718A, the resulting window 701 shown in **Figure 7E** then appears. There are several aspects of this user interface shown in **Figure 7A-7E** which are worthy of being noted. The search parameters or search query is specified within the same window as the display of the search results. This allows the user to look
at a single location or window to understand the search parameters and how they affected the displayed search results, and may make it easier for a user to alter or improve the search parameters in order to find one or more files. The configurable pull down menus, such as the add search parameter pull down menu, includes hierarchical pull down menus. An example of this is shown in Figure 7B in which the selection of the time criteria from the pull down menu 717 results in the display of another menu, in this case a submenu 719A which may be selected from by the user. This allows for a compact presentation of the various search parameters while keeping the initial complexity (e.g. without submenus being displayed) at a lower level. Another useful aspect of the user interface shown in Figure 7A-7E is the ability to reconfigure pull down menus which have previously been configured. Thus, for example, the configurable pull down menu 713 currently specifies the location of the search (in this case, all local disks), however, this may be modified by selecting the pull down region associated with the configurable pull down menu 713, causing the display of a menu of options indicating alternative locations which may be selected by the user. This can also be seen in Figure 7D in which the past week option has been selected by the user (as indicated by "past week" being in the search parameter menu bar 707), but a menu of options shown in the pull down menu 718A allows the user to change the selected time from the "past week" to some other time criteria. Another useful aspect of this user interface is the ability to continue adding various search criteria by using the add search criteria pull down menu 717 and selecting a new criteria.

[0070] It will also be appreciated that the various options in the pull down menus may depend upon the fields within a particular type of metadata file. For example, the selection of "images" to be searched may cause the various fields present in the metadata for an image type file to appear in one or more pull down menus, allowing the user to search within one or more of those fields for that particular type of file. Other fields which do not apply to "images" types of files
may not appear in these menus in order reduce the complexity of the menus and to prevent user confusion.

[0071] Another feature of the present invention is shown in Figures 7A-7E. In particular, the side bar region 703A, which is the user-configurable portion of the side bar, includes a representation of a folder 725 which represents the search results obtained from a particular search, which search results may be static or they may be dynamic in that, in certain instances, the search can be performed again to obtain results based on the current files in the system. The folder 725 in the example shown in Figures 7A-7E represents a search on a local disk for all images done on December 10th. By selecting this folder in the side bar region 703A, the user may cause the display in the display region 705 of the results of that search. In this way, a user may retrieve a search result automatically by saving the search result into the side bar region 703A. One mechanism for causing a search result or a search query to be saved into the side bar region 703A is to select the add folder button 727 which appears in the bottom portion of the window 701. By selecting this button, the current search result or search query is saved as a list of files and other objects retrieved in the current search result. In the case where the search query is saved for later use rather than the saving of a search result, then the current search query is saved for re-use at a later time in order to find files which match the search query at that later time. The user may select between these two functionalities (saving a search result or saving a search query) by the selection of a command which is not shown.

[0072] Figures 8A and 8B show another aspect of a user interface feature which may be used with certain embodiments of the present invention. The window 801 of Figure 8A represents a display of the search results which may be obtained as a result of using one of the various different embodiments of the present invention. The search results are separated into categories which are separated by headers 805, 807, 809, and 811 which in this case represent periods of time. This particular segmentation with headers was selected by the user's
selecting the heading "date modified" using the date modified button 803 at the top of the window 801. An alternative selection of the kind category by selecting the button 802 at the top of the window 801A shown in Figure 8B results in a different formatting of the search results which are now categorized by headers which indicate the types of files which were retrieved in the search and are separated by the headings 815, 817, 819, and 821 as shown in Figure 8B. The use of these headings in the search results display allows the user to quickly scan through the search results in order to find the file.

[0073] Figure 9 shows another aspect of the present invention that is illustrated as part of the window 901 shown in Figure 9. This window includes a display region 905 which shows the results of the search and the window also includes two side bar regions 903A and 903B, where the side bar region 903A is the user-configurable portion and the side bar region 903B is the system controlled portion. A folder add button 927 may be selected by the user to cause the addition of a search result or a search query to be added to the user-configurable portion of the side bar. The window 901 also includes conventional window controls such as a title bar or region 929 which may be used to move the window around a display and view select buttons 937 and maximize, minimize and resize buttons 934, 935, and 936 respectively. The window 901 shows a particular manner in which the results of a text-based search may be displayed. A text entry region 909 is used to enter text for searching. This text may be used to search through the metadata files or the indexed files or a combination of both. The display region 905 shows the results of a search for text and includes at least two columns, 917 and 919, which provide the name of the file that was found and the basis for the match. As shown in column 919, the basis for the match may be the author field or a file name or a key word or comments or other data fields contained in metadata that was searched. The column 921 shows the text that was found which matches the search parameter typed into the text entry field 909. Another column 911 provides additional information with respect to the search
results. In particular, this column includes the number of matches for each particular type of category or field as well as the total number of matches indicated in the entry 913. Thus, for example, the total number of matches found for the comments field is only 1, while other fields have a higher number of matches.

[0074] Figure 10 shows certain other aspects of some embodiments of the present invention. Window 1001 is another search result window which includes various fields and menus for a user to select various search parameters or form a search query. The window 1001 includes a display region 1005 which may be used to display the results of a search and a user-configurable side bar portion 1003A and a system specified side bar portion 1003B. In addition, the window 1001 includes conventional scrolling controls such as controls 1021 and 1022 and 1021A. The window further includes conventional controls such as a title bar 1029 which may be used to move the window and view control buttons 1037 and maximize, minimize, and resize buttons 1034, 1035, and 1036. A start search button 1015 is near a text entry region 1009. A first search parameter menu bar 1007 is displayed adjacent to a second search parameter bar 1011. The first search parameter search bar 1007 allows a user to specify the location for a particular search while two menu pull down controls in the second search parameter menu bar 1011 allow the user to specify the type of file using the pull down menu 1012 and the time the file was created or last modified using the menu 1013.

[0075] The window 1001 includes an additional feature which may be very useful while analyzing a search result. A user may select individual files from within the display region 1005 and associate them together as one collection. Each file may be individually marked using a specific command (e.g. pressing the right button on a mouse and selecting a command from a menu which appears on the screen, which command may be "add selection to current group") or similar such commands. By individually selecting such files or by selecting a group of
files at once, the user may associate this group of files into a selected group or a "marked" group and this association may be used to perform a common action on all of the files in the group (e.g. print each file or view each file in a viewer window or move each file to a new or existing folder, etc.). A representation of this marked group appears as a folder in the user-configurable portion 1003A. An example of such a folder is the folder 1020 shown in the user-configurable portion 1003A. By selecting this folder (e.g. by positioning a cursor over the folder 1020 and pressing and releasing a mouse button or by pressing another button) the user, as a result of this selection, will cause the display within the display region 1005 of the files which have been grouped together or marked. Alternatively, a separate window may appear showing only the items which have been marked or grouped. This association or grouping may be merely temporary or it may be made permanent by retaining a list of all the files which have been grouped and by keeping a folder 1020 or other representations of the grouping within the user-configurable side bar, such as the side bar 1003A. Certain embodiments may allow multiple, different groupings to exist at the same time, and each of these groupings or associations may be merely temporary (e.g. they exist only while the search results window is displayed), or they may be made permanent by retaining a list of all the files which have been grouped within each separate group. It will be appreciated that the files within each group may have been created from different applications. As noted above, one of the groupings may be selected and then a user may select a command which performs a common action (e.g. print or view or move or delete) on all of the files within the selected group.

[0076] Figures 11A, 11B, 11C, and 11D show an alternative user interface for allowing a user to input search queries or search parameters. The user interface shown in these figures appears within the window 1101 which includes a user-configurable side bar region 1103A and a system specified side bar region 1103B. The window 1101 also includes traditional window controls such as a
window resizing control 1131 which may be dragged in a conventional graphical user interface manner to resize the window, and the window further includes scrolling controls such as controls 1121, 1122, and 1123. The scrolling control 1121 may, for example, be dragged within the scrolling region 1121A or a scroll wheel on a mouse or other input device may be used to cause scrolling within a display region 1105. Further, traditional window controls include the title bar 1129 which may be used to move the window around a desktop which is displayed on a display device of a computer system and the window also includes view buttons 1137 as well as close, minimize, and resize buttons 1134, 1135 and 1136. A back and forward button, such as the back button 1132, are also provided to allow the user to move back and forth in a manner which is similar to the back and forth commands in a web browser. The window 1101 includes a search parameter menu bar 1111 which includes a "search by" pull down menu 1112 and a "sort by" pull down menu 1114. The "search by" pull down menu 1112 allows a user to specify the particular search parameter by selecting from the options which appear in the pull down menu once it is activated as shown in Figure 11B. In particular, the pull down menu 1113 shows one example of a pull down menu when the "search by" pull down menu 1112 has been activated. The "sort by" pull down menu 1114 allows a user to specify how the search results are displayed within a display region 1105. In the example shown in Figures 11A-11D a user has used the "sort by" pull down menu 1114 to select the "date viewed" criteria to sort the search results by. It should also be noted that the user may change the type of view of the search results by selecting one of the three view buttons 1137. For example, a user may select an icon view which is the currently selected button among the view buttons 1137, or the user may select a list view or a column view.

[0077] Figure 11B shows the result of the user's activation of a "search by" pull down menu 1112 which causes the display of the menu 1113 which includes a plurality of options from which the user may choose to perform a search by. It
will be appreciated that there are a number of different ways for a user to activate the "search by" pull down menu 1112. One way includes the use of a cursor, such as a pointer on a display which is controlled by a cursor control device, such as a mouse. The cursor is positioned over the region associated with the "search by" menu title (which is the portion within the search parameter menu bar 1111 which contains the words "search by") and then the user indicates the selection of the menu title by pressing a button, such as a mouse's button, to cause the pull down menu to appear, which in this case is the menu 1113 shown in Figure 11B. At this point, the user may continue to move the cursor to point to a particular option within the menu, such as the "time" option. This may result in the display of a submenu to the left or to the right of the menu 1113. This submenu may be similar to the submenu 719A or to the menu 1214 shown in Figure 12A. If the "kind" option is selected in the menu 1113, the submenu may include a generic list of the different kinds of documents, such as images, photos, movies, text, music, PDF documents, email documents, etc. or the list may include references to specific program names such as PhotoShop, Director, Excel, Word, etc. or it may include a combination of generic names and specific names. Figure 11C shows the result of the user having selected PhotoShop type of documents from a submenu of the "kind" option shown in menu 1113. This results in the display of the search parameter menu bar 1111A shown in Figure 11C which includes a highlighted selection 1111B which indicates that the PhotoShop type of documents will be searched for. The search parameter menu bar 1111 appears below the search parameter menu bar 1111A as shown in Figure 11C. The user may then specify additional search parameters by again using the "search by" pull down menu 1112 or by typing text into the text entry field 1109. For example, from the state of the window 1101 shown in Figure 11C, the user may select the "search by" pull down menu 1112 causing the display of a menu containing a plurality of options, such as the options shown within the menu 1113 or alternative options such as those which relate to PhotoShop documents (e.g. the
various fields in the metadata for PhotoShop type of documents). A combination of such fields contained within metadata for PhotoShop type documents and other generic fields (e.g. time, file size, and other parameters) may appear in a menu, such as the menu 1113 which is activated by selecting the "search by" pull down menu. The user may then select another criteria such as the time criteria. In this case, the window 1101 displays a new search parameter menu bar 1115 which allows a user to specify a particular time. The user may select one of the times on the menu bar 1115 or may activate a pull down menu by selecting the menu title "time," which is shown as the menu title 1116. The state of the window 1101 shown in Figure 11D would then search for all PhotoShop documents created in the last 30 days or 7 days or 2 days or today or at any time, depending on the particular time period selected by the user.

[0078] Figures 12A, 12B, 12C and 12D show another example of a user interface for allowing the creation of search queries for searching metadata and other data and for displaying the results of the search performed using a search query. The different implementation shown in Figures 12A-12D shows a user interface presentation in a column mode; this can be seen by noting the selection of the column button, which is the rightmost button in the view buttons 1237 shown in Figure 12A. The window 1201 has two columns 1211 and the display region 1205, while the window 1251 of Figure 12C has three columns which are columns 1257, 1259, and the display region 1255, and the window 1271 has three columns which are columns 1277, 1279, and the display region 1275.

[0079] The window 1201 shown in Figures 12A and 12B includes a display region 1205 which shows the results of a search; these results may be shown dynamically as the user enters search parameters or the results may be shown only after the user has instructed the system to perform the search (e.g. by selecting a "perform search" command). The window 1201 includes conventional window controls, such as a resizing control 1231, a scrolling control 1221, a title bar 1229 which may be used to move the window, a window close button, a window
minimize button, and a window resize button 1234, 1235, and 1236, respectively. The window 1201 also includes a user-configurable side bar region 1203A and a system specified side bar region 1203B. It can be seen from Figure 12A that a browse mode has been selected as indicated by the highlighted "browse" icon 1203C in the system specified side bar region 1203B. The window 1201 also includes a text entry region 1209, which a user may use to enter text for a search, and the window 1201 also includes view selector buttons 1237.

[0080] A column 1211 of window 1201 allows a user to select various search parameters by selecting one of the options which in turn causes the display of a submenu that corresponds to the selected option. In the case of Figure 12A, the user has selected the "kind" option 1212 and then has used the submenu 1214 to select the "photos" option from the submenu, resulting in an indicator 1213 (photos) to appear in the column 1211 under the "kind" option as shown in Figure 12A. It can also be seen that the user has previously selected the "time" option in the column 1211 and has selected from a submenu brought up when the "time" option was selected the "past week" search parameter. When the user has finished making selections of the various options and suboptions from both the column 1112 and any of the corresponding submenus which appear, then the display showed in Figure 12B appears. Note that the submenus are no longer present and that the user has completed the selection of the various options and suboptions which specify the search parameters. Column 1211 in Figure 12B provides feedback to the user indicating the exact nature of the search query (in this case a search for all photos dated in the past week), and the results which match the search query are shown in the display region 1205.

[0081] Figures 12C and 12D show an alternative embodiment in which the submenus which appear on a temporary basis in the embodiment of Figures 12A and 12B are replaced by an additional column which does not disappear after a selection is made. In particular, the column 1259 of the window 1251 functions in the same manner as the submenu 1214 except that it remains within the
window 1251 after a selection is made (wherein the submenu 1214 is removed from the window after the user makes the selection from the submenu). The column 1279 of window 1271 of Figure 12D is similar to the column 1259. The window 1251 includes a side bar which has a user-configurable side bar region 1253A and a system defined side bar region 1253B. The system specified side bar region 1253B includes a "browse" selection region 1254 which has a clear button 1258 which the user may select to clear the current search query. The window 1271 of Figure 12D provides an alternative interface for clearing the search query. The window 1271 also includes a user configurable side bar region 1273A and a system specified side bar region 1273B, but the clear button, rather than being with the "search" region 1274 is at the top of the column 1277. The user may clear the current search parameter by selecting the button 1283 as shown in Figure 12D.

[0082] Figure 13A shows another embodiment of a window 1301 which displays search results within a display region 1302. The window 1301 may be a closeable, minimizeable, resizeable, and moveable window having a resizing control 1310, a title bar 1305 which may be used to move the window, a text entry region 1306 and a user configurable portion 1303, and a system specified portion 1304. The window 1301 further includes buttons for selecting various views, including an icon view, a list view, and a column view. Currently, the list view button 1316 has been selected, causing the display of the search results in a list view manner within the display region 1302. It can be seen that the text ("button") has been entered into the text entry region 1306 and this has caused the system to respond with the search results shown in the display region 1302. The user has specified a search in every location by selecting "everywhere" button 1317. Further, the user has searched for any kind of document by selecting the "kind" option from the pull down menu 1315 and by selecting the "any" option in the pull down menu 1319. The where or location slice 1307 includes a "+" button which may be used to add further search parameters, and similarly, the slice 1308
includes a "+" and a "-" button for adding or deleting search parameters, respectively. The slice 1307 further includes a "save" button 1309 which causes the current search query to be saved in the form of a folder which is added to the user configurable portion 1303 for use later. This is described further below and may be referred to as a "smart folder." The search input user interface shown in Figures 13A and 13B is available within, in certain embodiments, each and every window controlled by a graphical user interface file management system, such as a Finder program which runs on the Macintosh or Windows Explorer which runs on Microsoft Windows. This interface includes the text entry region 1306 as well as the slices 1307 and 1308.

[0083] The window 1301 shown in Figure 13B shows the activation of a menu by selecting the search button 1323A, causing a display of a menu having two entries 1323 and 1325. Entry 1323 displays recently performed searches so that a user may merely recall a prior search by selecting the prior search and cause the prior search to be run again. The menu selection 1325 allows the user to clear the list of recent searches in the menu.

[0084] Figures 14A, 14B, and 14C show examples of another window in a graphical user interface file system, such as the Finder which runs on the Macintosh operating system. These windows show the results of a particular search and also the ability to save and use a smart folder which saves a prior search. The window 1401 shown in Figure 14A includes a display region 1403, a user configurable region 1405, a smart folder 1406, a system specified region 1407, an icon view button 1409, a list view button 1410, and a column view button 1411. The window 1401 also includes a text entry region 1415 and a location slice 1416 which may be used to specify the location for the search, which slice also includes a save button 1417. Additional slices below the slice 1416 allow the user to specify further details with respect to the search, in this case specifying types of documents which are images which were last viewed this week. The user has set the search parameters in this manner by selecting the
"kind" option from the pull down menu 1419 and by selecting the "images" type from the pull down menu 1420 and by selecting the "last viewed" option from pull down menu 1418 and by selecting "this week" from the pull down menu 1422. The user has also selected "everywhere" by selecting the button 1421 so that the search will be performed on all disks and storage devices connected to this system. The results are shown within the display region 1403. The user can then save the search query by selecting the "save" button 1417 and may name the saved search query as "this week's images" to produce the smart folder 1406 as shown in the user configurable portion 1405. This allows the user to repeat this search at a later time by merely selecting the smart folder 1406 which causes the system to perform a new search again, and all data which matches the search criteria will be displayed within the display region 1403. Thus, after several weeks, a repeating of this search by selecting the smart folder 1406 will produce an entirely different list if none of the files displayed in the display region 1403 of Figure 14A are viewed in the last week from the time in which the next search is performed by selecting the smart folder 1406.

[0085] Figure 14B shows a way in which a user may sort or further search within the search results specified by a saved search, such as a smart folder. In the case of Figure 14B, the user has selected the smart folder 1406 and has then entered text "jpg" 1425 in the text entry region 1415. This has caused the system to filter or further limit the search results obtained from the search query saved as the smart folder 1406. Thus, PhotoShop files and other files such as TIF files and GIF files are excluded from the search results displayed within the display region 1403 of Figure 14B because the user has excluded those files by adding an additional search criteria specified by the text 1425 in the text entry region 1415. It can be seen that the "jpg" text entry is ANDed logically with the other search parameters to achieve the search results displayed in the display region 1403. It can also be seen that the user has selected the icon view by selecting the icon view button 1409. Thus, it is possible for a user to save a search query and use it
later and to further limit the results of the search query by performing a search on
the results of the search query to further limit the search results.

[0086] **Figure 14C** shows the window 1401 and shows the search results
displayed within the display region 1403, where the results are based upon the
saved search specified by the smart folder 1406. The user has caused a pull down
menu 1427 to appear by selecting the pull down region 1427A. The pull down
region 1427 includes several options which a user may select. These options
include hiding the search criteria or saving the search (which is similar to
selecting the button 1417) or showing view options or opening the selected file.
This allows the user, for example, to hide the search criteria, thereby causing the
slice 1416 and the other search parameters to be removed from the window 1401
which is a moveable, resizeable, minimizeable, and closeable window.

[0087] **Figure 14D** shows an example of a user interface which allows the
user to specify the appearance of a smart folder, such as the smart folder 1406.

[0088] **Figure 15A, 15B, 15C, and 15D** show an example of a system wide
search input user interface and search result user interface. In one particular
exemplary embodiment, these user interfaces are available on the entire system
for all applications which run on the system and all files and metadata, and even
address book entries within an address book program, such as a personal
information manager, and calendar entries within a calendar program, and emails
within an email program, etc. In one exemplary embodiment, the system begins
performing the search and begins displaying the results of the search as the user
types text into a text entry field, such as the text entry field 1507. The search
results are organized by categories and are displayed as a short list which is
intentionally abbreviated in order to present only a selected number of the most
relevant (scored) matches or hits to the search query. The user can ask for the
display of all the hits by selecting a command, such as the "show all" command
1509. **Figure 15A** shows a portion of a display controlled by a data processing
system. This portion includes a menu bar 1502 which has at its far end a search
menu command 1505. The user can select the search menu command by
positioning a cursor, using a mouse, for example, over the search menu command
1505 and by pressing a button or by otherwise activating or selecting a command.
This causes a display of a text entry region 1507 into which a user can enter text.
In the example shown in Figure 15A, which is a portion of the display, the user
has entered the text "shakeit" causing the display of a search result region
immediately below a "show all" command region 1509 which is itself
immediately below the text entry region 1507. It can be seen that the hits or
matches are grouped into categories ("documents" and "PDF documents") shown
by categories 1511 and 1513 within the search result region 1503. Figure 15B
shows another example of a search. In this case, a large number of hits was
obtained (392 hits), only a few of which are shown in the search result region
1503. Again, the hits are organized by categories 1511 and 1513. Each category
may be restricted in terms of the number of items displayed within the search
result region 1503 in order to permit the display of multiple categories at the same
time within the search result region. For example, the number of hits in the
documents category may greatly exceed the available display space within the
search result region 1503, but the hits for this category are limited to a
predetermined or dynamically determinable number of entries within the search
result region 1503 for the category 1511. An additional category, "top hit" is
selected based on a scoring or relevancy using techniques which are known in the
art. The user may select the "show all" command 1509 causing the display of a
window, such as window 1601 shown in Figure 16A. Figure 15C shows a
display of a graphical user interface of one embodiment of the invention which
includes the menu bar 1502 and the search menu command 1505 on the menu bar
1502. Figure 15D shows another example of the search result region 1503 which
appeared after a search of the term "safari" was entered into the text entry region
1507. It can be seen from the search result region 1503 of Figure 15D that the
search results are again grouped into categories. Another search result window
1520 is also shown in the user interface of Figure 15D. It can be seen that application programs are retrieved as part of the search results, and a user may launch any one of these application programs by selecting it from the search result region, thereby causing the program to be launched.

[0089] Figures 16A and 16B show examples of search result windows which may be caused to appear by selecting the "show all" command 1509 in Figures 15A or 15B. Alternatively, these windows may appear as a result of the user having selected a "find" command or a some other command indicating that a search is desired. Moreover, the window 1601 shown in Figures 16A and 16B may appear in response to either of the selection of a show all command or the selection of a find command. The window 1601 includes a text entry region 1603, a group by menu selection region 1605, a sort by menu selection region 1607, and a where menu selection region 1609. The group by selection region 1605 allows a user to specify the manner in which the items in the search results are grouped according to. In the example shown in Figure 16A, the user has selected the "kind" option from the group by menu selection region 1605, causing the search results to be grouped or sorted according to the kind or type of document or file. It can be seen that the type of file includes "html" files, image files, PDF files, source code files, and other types of files as shown in Figure 16A. Each type or kind of document is separated from the other documents by being grouped within a section and separated by headers from the other sections. Thus, headers 1611, 1613, 1615, 1617, 1619, 1621, and 1623 designate each of the groups and separate one group from the other groups. This allows a user to focus on evaluating the search results according to certain types of documents. Within each group, such as the document groups or the folder groups, the user has specified that the items are to be sorted by date, because the user has selected the date option within the sort by menu region 1607. The user has also specified that all storage locations are to be searched by selecting "everywhere" from the where menu selection region 1609. Each item in the search result list includes an
information button 1627 which may be selected to produce the display of additional information which may be available from the system. An example of such additional information is shown in Figure 17 in which a user has selected the information button 1627 for item 1635, resulting in the display of an image 1636 corresponding to the item as well as additional information 1637. Similarly, the user has selected the information button for another item 1630 to produce the display of an image of the item 1631 as well as additional information 1632. The user may remove this additional information from the display by selecting the close button 1628 which causes the display of the information for item 1635 to revert to the appearance for that item shown in Figure 16A. The user may collapse an entire group to hide the entries or search results from that group by selecting the collapse button 1614 shown in Figure 16A, thereby causing the disappearance of the entries in this group as shown in Figure 16B. The user may cause these items to reappear by selecting the expand button 1614A as shown in Figure 16B to thereby revert to the display of the items as shown in Figure 16A.

[0090] The search results user interface shown in Figures 16A and 16B presents only a limited number of matches or hits within each category. In the particular example of these figures, only the five top (most relevant or most highly sorted) hits are displayed. This can be seen by noticing the entry at the bottom of each list within a group which specifies how many more hits are within that group; these hits can be examined by selecting this indicator, such as indicator 1612, which causes the display of all of the items in the documents category or kind for the search for "button" which was entered into the text entry region 1603. Further examples of this behavior are described below and are shown in conjunction with Figures 18A and 18B. It will be appreciated that window 1601 is a closeable and resizable and moveable window and includes a close button and a resizing control 1625A.

[0091] Figures 18A and 18B illustrate another window 1801 which is very similar to the window 1601. The window 1801 includes a text entry region 1803,
a group by menu selection region 1805, a sort by menu selection region 1807, and a
where menu selection region 1809, each of which function in a manner which is
similar to the regions 1605, 1607, and 1609 respectively of Figure 16A. Each
item in a list view within the window 1801 includes an information button 1827,
allowing a user to obtain additional information beyond that listed for each item
shown in the window 1801. The window 1801 further includes headers 1811,
1813, 1815, 1817, 1819, 1821, and 1823 which separate each group of items,
grouped by the type or kind of document, and sorted within each group by date,
from the other groups. A collapse button 1814 is available for each of the
headers. The embodiment shown in Figures 18A and 18B shows the ability to
switch between several modes of viewing the information. For example, the user
can display all of the hits within a particular group by selecting the indicator
1812 shown in Figure 18A which results in the display of all of the images files
within the window 1801 within the region 1818A. The window is scrollable,
thereby allowing the user to scroll through all the images. The user can revert
back to the listing of only five of the most relevant images by selecting the "show
top 5" button 1832 shown in Figure 18B. Further, the user can select between a
list view or an icon view for the images portion shown in Figures 18A and 18B.
The user may select the list view by selecting the list view button 1830 or may
select the icon view by selecting the icon view button 1831. The list view for the
images group is shown in Figure 16A and the icon view for the images group is
shown in Figures 18A and 18B. It can be seen that within a single, moveable,
resizable, closeable search result window, that there are two different views (e.g.
a list view and an icon view) which are concurrently shown within the window.
For example, the PDF documents under the header 1819 are displayed in a list
view while the images under the header 1817 are displayed in an icon view in
Figures 18A and 18B. It can also be seen from Figures 18A and 18B that each
image is shown with a preview which may be capable of live resizing as
described in a patent application entitled "Live Content Resizing" by inventors.
Steve Jobs, Steve Lemay, Jessica Kahn, Sarah Wilkin, David Hyatt, Jens Alfke, Wayne Loofbourrow, and Bertrand Serlet, filed on the same date as this application, and being assigned to the assignee of the present inventions described herein, and which is hereby incorporated herein by reference.

Figure 19A shows another example of a search result window which is similar to the window 1601. The window 1901 shown in Figure 19A includes a text entry region 1903 and a group by menu selection region 1905 and a sort by menu selection region 1907 and a where menu selection region 1908. Further, the window includes a close button 1925 and a resizing control 1925A. Text has been entered into the text entry region 1903 to produce the search results shown in the window 1901. The search results again are grouped by a category selected by a user which in this case is the people options 1906. This causes the headers 1911, 1913, 1915, and 1917 to show the separation of the groups according to names of people. Within each group, the user has selected to sort by the date of the particular file or document. The user interface shown in Figure 19A allows a user to specify an individual's name and to group by people to look for communications between two people, for example. Figure 19B shows another way in which a user can group a text search ("imran") in a manner which is different from that shown in Figure 19A. In the case of Figure 19B, the user has selected a flat list from the group by menu selection region 1905 and has selected "people" from the sort by menu region 1907. The resulting display in window 1901A is without headers and thus it appears as a flat list.

Figure 19C shows the user interface of another search result window 1930 which includes a text entry region 1903 and the selection regions 1905, 1907, and 1908 along with a scrolling control 1926. The results shown in the window 1930 have been grouped by date and sorted within each group by date. Thus, the headers 1932, 1934, 1936, 1938, and 1940 specify time periods such as when the document was last modified (e.g. last modified today, or yesterday, or last week). Also shown within the search results window 1930 is the information
button 1942 which may be selected to reveal further information, such as an icon 1945 and additional information 1946 as shown for one entry under the today group. This additional information may be removed by selecting the contraction button 1944.

[0094] **Figure 19D** shows a search result window 1950 in which a search for the text string "te" is grouped by date but the search was limited to a "home" folder as specified in the where menu selection region 1908. Time specific headers 1952, 1954, 1956, and 1958 separate items within one group from the other groups as shown in **Figure 19D**.

[0095] **Figure 19E** shows an alternative embodiment of a search result window. In this embodiment, the window 1970 includes elements which are similar to window 1901 such as the selection regions 1905, 1907, and a scrolling control 1926 as well as a close button 1925 and a resizing control 1925A. The search result window 1970 further includes a "when" menu selection region 1972 which allows the user to specify a search parameter based on time in addition to the text entered into the text entry region 1903. It can be seen from the example shown in **Figure 19E** that the user has decided to group the search results by the category and to sort within each group by date. This results in the headers 1973, 1975, 1977, and 1979 as shown in **Figure 19E**.

[0096] **Figure 20** shows an exemplary method of operating a system wide menu for inputting search queries, such as the system wide menu available by selecting the search menu command 1505 shown in **Figure 15A** or **15B**, or **15C**. In operation 2001, the system displays a system wide menu for inputting search queries. This may be the search menu command 1505. The user, in operation 2003, inputs a search, and as the search query is being inputted, the system begins performing and begins displaying the search results before the user finishes inputting the search query. This gives immediate feedback and input to the user as the user enters this information. The system is, in operation 2005, performing a search through files, metadata for the files, emails within an email program,
address book entries within an address book program, calendar entries within a calendar program, etc. The system then, in operation 2007, displays an abbreviated (e.g. incomplete) list of hits if there are more than a certain number of hits. An example of this abbreviated listing is shown in Figure 15B. The listing may be sorted by relevance and segregated into groups such as categories or types of documents. Then in operation 2009, the system receives a command from the user to display all the hits and in operation 2011 the system displays the search results window, such as the window 1601 shown in Figure 16A. This window may have the ability to display two different types of views, such as an icon view and a list view within the same closeable, resizable, and moveable window. It will be appreciated that the searching, which is performed as the user is typing and the displaying of results as the user is typing may include the searching through the metadata files created from metadata extracted from files created by many different types of software programs.

[0097] Figures 21, and 22A, 22B, 22C, and 22D will now be referred to while describing another aspect of the inventions. This aspect relates to a method of selecting a group of files, such as a group of individual data files. In an exemplary method of this aspect, a data processing system receives a selection of a plurality of items, such as data files, folders (e.g. graphical user interface representations of subdirectories), application programs or a combination of one or more of these items. This selection may be performed by one of the many conventional ways to select a plurality of items such as (a) positioning a cursor at each item individually (e.g. through the movement of a mouse) and indicating a selection individually by, for example, pressing and releasing a button, such as a mouse's button; (b) pointing a cursor at a first item in a list and indicating a selection of the first item and pointing the cursor at a last item in a list of items and indicating a selection of all items from the first item to the last item in the list; (c) drawing a selection rectangle by a dragging operation of the cursor, etc. Thus operation 2101 shown in Figure 21 receives one or more inputs indicating a
selection of a plurality of items. The system in operation 2103 receives a command requesting both the creation of a new storage facility (e.g. a folder) and an association of the plurality of items with the new storage facility. While the operation 2103 is shown following operation 2101, in certain embodiments operation 2103 may precede operation 2101. The association of operation 2103 may be a copy or a move operation. For example, the user may select multiple items and then command the system to move those items from their existing locations to a new folder which is created in one operation as a result of the move and create new folder command. In response to the command received in operation 2103, the system creates a new storage facility, such as a new folder, with a predetermined directory path name or a user specified path name and the system further associates the selected plurality of items with the new storage facility. This association may be either a move or a copy operation. A copy operation would typically involve making a copy of each selected item and storing the item with a path name that reflects the storage of the item within the new folder having a predetermined directory path name or a user specified directory path name. A move operation, in which the items are moved into the new folder, may merely change the path names associated with each of the selected items (rather than making a copy of the items) which changed path names will reflect the new file system location (e.g. within the subdirectory of the new folder) of the selected items.

[0098] Figures 22A-22D show one example of the method of Figure 21. A desktop 2201 on a display device is shown containing multiple windows and also an icon 2227 on the desktop. A cursor 2211 is also shown on the desktop. The windows 2203, 2205, and 2207 each contain a plurality of items shown as icons. In particular, window 2203 includes a data file represented by icon 2215 in a folder (e.g. a graphical representation of a subdirectory in a file storage system) represented by icon 2217. The window 2205 includes a program icon 2223 and a document icon 2219 and another document icon 2225 and a folder icon 2221.
The window 2207 shows a list view of several files including "File B." The user may then, using the cursor 2211 or using other conventional user interface techniques, select multiple items. This may be done with one input or more inputs which indicate the selection of multiple items. Figure 22B shows the result of the user having selected icons 2215, 2217, 2223, 2225, 2227, and "File B" in window 2207. It can be seen that the cursor 2211 is positioned adjacent to the icon 2225 at this point in the operation. Then the user, after having selected a plurality of items, may invoke the command referred to in operation 2103. An example of this is shown in Figure 22C which represents a portion of the desktop 2101, which portion is designated 2201A as shown in Figure 22C. The user has caused a pop up menu 2230 to appear, which pop up menu includes three options 2231, 2232, and 2233. Option 2231 would allow a user to move all the selected items into the trash (e.g. delete them) while options 2232 and 2233 relate to the command referred to in operation 2103 of Figure 21. In particular, option 2232 is a command which is selectable by the user to create a new folder and, in the same operation, move the items which have been selected into the new folder. Option 2233 is a command which allows the user to, in one operation, create a new folder and copy the selected items into the new folder. In the example shown in Figures 22A-22D, the user will select option 2232, thereby causing the system to create a new storage facility, such as a new folder with a predetermined directory name (e.g. "new folder") or alternatively, a user specified path name. This result is shown in Figure 22D in which the desktop 2201 now includes a new window labeled "new folder" which represents and shows the contents of this new folder, which is also shown as the folder 2253 which is a graphical user interface representation of this new folder.

[0099] It will be appreciated that this method may employ various alternatives. For example, a window may appear after the command option 2232 or 2233 has been selected, and this window asks for a name for the new folder. This window may display a default name (e.g. "new folder") in case the user does
not enter a new name. Alternatively, the system may merely give the new folder or new storage facility a default path name. Also, the system may merely create the new folder and move or copy the items into the new folder without showing the new window as shown in Figure 22D.

[00100]  Figure 23 relates to an aspect of processing information in order to update one or more databases, such as an index database or a metadata database. The updating of the one or more databases is performed and controlled in response to user activity on the data processing system. Typically, higher use by the user will cause the system to automatically adjust the amount of updating of either the index database or the metadata database so that the user’s activities and use of the data processing system are not adversely impacted by the updating operations on the one or more databases. It will be appreciated that in certain embodiments, only one type of updating, rather than both types of updating, is controlled in response to user activity; for example, since indexing is often more demanding of a computer’s resources, the indexing may be controlled in response to user activity, but the operations for the metadata database are not. Operation 2301 involves the monitoring of the user’s use of the data processing system. This may be accomplished in a variety of different ways, depending upon the data processing system. For example, data processing systems which use a UNIX based operating system typically include a facility which provides an indication of the amount of use of programs which are “owned” by the user as opposed to programs which are controlled or owned by the data processing system, which may be referred to as the “root” user. One or more thresholds of user activity may be used to determine whether a user’s use or activity on the data processing system is at a higher level or a lower level. For example, if the user is not entering data or positioning a cursor and programs owned by the user are not performing computations or processing data, then typically, user activity will be considered to be low. On the other hand, if the user is entering data or causing data to be entered or has caused one of the user’s programs to perform a complex
set of computations, then the data processing system will normally determine that the user's use of the system is high. Operation 2303 shows how the system responds in one exemplary embodiment when the user's use is at a higher level. In this case, the data processing system either stops indexing and/or importing of metadata operations or sets the indexing and/or metadata software to a lower priority (e.g. the Nice command is changed), such as a lower processing priority for one or both of the indexing software and the metadata software. This will cause the system to provide more processing time to the user's programs and less processing time to the indexing software and/or metadata software. This type of control is useful in an operating system which provides preemptive multitasking. The data processing system will typically operate in a loop as shown in Figure 23 in which, after it adjusts processing priorities, it will continue to monitor the user's use. If the system determines that the user's use is below a threshold level, then the system may give the indexing and/or metadata software a higher processing priority, as shown in operation 2305 and then return to monitoring of the user's use in operation 2301.

[00101] Figure 24 shows a method according to another aspect of the inventions described herein. In this method, notifications are used to automatically cause updating of an index database. These notifications may optionally be entered into an index queue (e.g. a "to be indexed" queue), and these notifications are typically not based solely on time or solely on a user input. In operation 2403, a notification is automatically generated in response to saving changes to an existing file or saving a newly created file. For example, an operating system's kernel may automatically generate a notification in response to a user or system initiated save, such as writing the file to a hard drive. The system may automatically perform this save (e.g. certain word processing programs and other software allow a user to set an automatic save operation to occur after the passage of a predetermined period of time). The notification which is generated automatically is directed to an indexing software component
which receives a notification in operation 2405. This notification may optionally be entered into an index queue which is further discussed in connection to Figure 25. In operation 2407, the indexing software performs indexing of the file and stores indexing data into an index database. This indexing is in response to the notification rather than the occurrence of a designated time or a user command to perform indexing. Further, the notification specifies the file by an identifier.

[00102] Figure 25 shows a specific exemplary method for using notifications to perform indexing and to also enter the notifications into a queue (e.g. "indexing queue") which is saved to a non-volatile storage device, such as a hard drive of the data processing system. Further, this method shows how information about changes to the queue may be entered into a transaction log. The files and the index database containing the indexed content of the files and the index queue and the log may all be stored on the same non-volatile storage device. This method of Figure 25 may also be performed for notifications to update metadata in a metadata database; in particular, the notifications to add metadata or change metadata in a metadata database may be added to a queue for updating the metadata and this queue (e.g. "metadata queue") may be stored in a non-volatile storage, and information about changes to this queue may be entered into the transaction log. The files and the metadata database containing metadata for the files and the metadata queue and the log may all be stored on the same non-volatile storage device. Typically, the log which includes the information about the changes to the queue is saved to the nonvolatile storage before the changes to the queue are saved to the nonvolatile storage.

[00103] The method of Figure 25 may begin in operation 2501 in which a notification from an operating system kernel is received. This notification indicates that an existing file has changed or a new file has been created. The notification normally includes an identifier which specifies the file (e.g. by its persistent, unique file ID number). In operation 2503, an identifier for the file is entered into an index queue and the queue is stored in a non-volatile storage, such
as a hard drive. Further, information about changes to the queue are entered into a transaction log maintained by the indexing software or the metadata database software. This log is designed to preserve the state of a data processing system through a crash of the system. Conventional techniques for implementing a log may be employed in order to preserve information about the queue. The log may be implemented as a simple log or as a log which is maintained as an “ACID” (Atomicity; Consistency; Isolation; and Durability) compliant journal which is similar to the journals maintained in journal file systems. Then in operation 2505, indexing is performed for files which are listed in the index queue. As the indexing software completes an indexing operation for a file, it causes the removal, in operation 2507, of an identifier for the file from the index queue and causes the storage of an updated queue in the non-volatile storage which also causes the entry of information about the changes to the queue into the transaction log.

[00104] Figure 25 shows a further aspect of certain embodiments. In particular, it shows how the system, through the use of the index queue and/or metadata queue can allow for the interruption of indexing or updating of metadata in response to increased user activity. This is shown in operations 2509 and 2511 which allow for the interruption of indexing in response to increased user use and then the resumption of indexing or metadata processing in response to decreased user use. The index queue and the metadata queue allow the system to keep track of changes to the files and enter information identifying those files in the index queue and metadata queue without having to perform indexing or importation of metadata while the user is actively using the data processing system.

[00105] Figure 26, 27, and 28 relate to yet another aspect of the inventions described herein. This aspect involves attempting to determine whether to update an index database or a metadata database without necessarily requiring an examination of each record or file indexed in the indexing database or added to the metadata database.
Figure 26 shows an exemplary method for determining whether to update a database, such as an index database or a metadata database. This method may be desirable in cases where the storage device is removable and can be moved from data processing system to data processing system, such as a USB hard drive or a USB flash drive and other types of removable storage devices or volumes. For example, a removable storage file may have been indexed on a first data processing system, and then the removable storage device is removed from the first data processing system and attached to a second data processing system which does not automatically perform indexing of content of files or the addition (e.g. importation) of metadata from files into a metadata database. A user may modify files on the storage volume while it is connected to the second data processing system without updating the index database or the metadata database. Also, the user may create new files and store them on the removable storage device while it is connected to the second data processing system. This can also occur without the content of those new files being added to the index database and without the metadata from those new files being added to the metadata database. Then, the removable storage device may be detached from the second data processing system and connected again to the first data processing system. The indexing software and metadata software on the first data processing system should attempt to determine whether the index database and/or the metadata database stored on the removable storage devices need to be updated. A similar need may arise even in the case of a storage device which is not removable. For example, a user may create and/or modify files in a flurry of activity which prevents the data processing system from obtaining the necessary processing time to index the new or modified files and to import metadata from the new or modified files. Immediately after the flurry of user activity, the user may instruct the system to shut down or power to the system may be lost (e.g. the system is powered by a battery, such as a laptop computer and insufficient charge is left on the battery, requiring the laptop computer to be shut down as soon as possible).
[00107] The method of Figure 26 may begin in operation 2601 in which it is determined when the file system's index database was last updated for a particular storage volume. In operation 2603, it is determined if any file was updated or created after the last update time of the index database. Optionally, rather than using the last update time of the index database, a period of time, such as one hour, earlier than that last update time may be utilized in order to compensate for differences in system clocks between one data processing system and another data processing system. Then in operation 2605, the data processing system updates the index database for any file updated or created after the last update time of the index database or, in the case where a period of time earlier than that last update time is used, such as one hour, the index database will be updated for any file updated or created after one hour earlier than the last update time of the index database. The technique of Figure 26 allows for determining whether to update the database based upon a time based comparison which does not require examining a record for each file which has been indexed in the indexing database. The technique of the method shown in Figure 26 may also be employed in determining whether or not to update a metadata database to add metadata from files which have been updated or created after the last time that the metadata database was updated (or after a period of time, such as one hour, earlier than that last time).

[00108] The method shown in Figure 27 shows a specific example of how to determine whether or not to update an index database or a metadata database. The method of Figure 27 may be performed in general using time comparisons or may be performed using the specific time comparisons shown in Figure 28. The method of Figures 27 and 28 may be used to determine whether to update an index database or a metadata database or both such databases which are stored on the volume. It will be appreciated that, at least in certain embodiments, the files which have been indexed and the metadata of those files which have been stored in the metadata database will be stored on the same device which also stores the
particular index database and the metadata database. That is, there is typically (in these embodiments) no separation of the index database from the files which have been indexed into that index database and similarly, there is no separation typically (in these embodiments) of the files and a metadata database containing metadata from those files. In an alternative embodiment, the database, such as index database or metadata database may be stored in a separate storage medium or storage device from the files from which the data in the database was obtained.

[00109] The method shown in Figure 27 may begin in operation 2701 in which the data processing system finishes indexing and closes the index database and index queue and un-mounts the volume which has been indexed (at least partially) by the first data processing system. Optionally, the first data processing system may, at shut down or on mounting of the volume perform operation 2801 shown in Figure 28. This operation involves recording three values which are the time that the index database was closed, the time that the file system on the volume was un-mounted and the write count for the volume at shut down or un-mounting. The volume may then be used in operation 2703 on another data processing system, and the files may possibly be modified on the volume without updating the index database and/or the metadata database. These databases may not be modified on data processing systems which do not have the necessary indexing software and/or metadata software. In operation 2705, the volume is again mounted on the first data processing system. Optionally, at this point in time, the first data processing system may perform operation 2803 which includes collecting three values which are the last mount time of the volume, the database open time, and the current write count. This can be seen in Figure 28. Then in operation 2707, the first data processing system determines whether any files on the volume have been modified after the index database was last closed and/or after the metadata database was last closed. Operation 2805 of Figure 28 shows one example of how to determine whether or not to update the databases. In operation 2805, it is determined whether the difference between two time values
T_{FS} and T_{DS} is less than a small delay in time, and similarly it is determined whether a difference between two other time values T_{Do} and T_{FM} is less than another small delay in time (which may be the same as or different than the other small delay in time). Further, the two write count values are compared and determined whether their difference is smaller than a small number. As shown in operation 2805 of Figure 28, if these three differences are less than certain values, then it is determined that the index database (and/or the metadata database) does not need to be updated. If any one of these comparisons fails the test in operation 2805, then it is determined that the database, such as the index database or the metadata database needs to be updated. The method of Figure 27 is useful even when the volume is not removable. It is often the case that a data processing system will be turned on and turned off repeatedly over a short period of time, such as one hour or less. It would be a waste of computing resources to scan files to determine whether to index new files. Thus the method of Figure 27 attempts to determine whether the databases need to be updated without examining a record for each file in the database. The specific method of Figure 28 attempts to verify that no file has been modified after the closing of the index database and does this by comparing several time values rather than by comparing time values of each file or by sorting the files by time.

[00110] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope of the invention as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.
CLAIMS

What is claimed is:

1. A machine implemented of processing data, the method comprising:
   mounting a storage device;
   determining, automatically in response to the mounting, whether to index
   one or more files on the storage device or to add metadata of the
   one or more files to a metadata database.

2. A method as in claim 1 wherein the determining comprises comparing a
   last time of usage or closing of an indexing database to an unmounting time or
   last write time of the storage device and also comprises comparing a first time of
   usage or opening of an indexing database after the unmounting time to a
   mounting time or first write time of the storage device.

3. A method as in claim 1 wherein the determining comprises evaluating
   whether files on the storage device have been modified since an indexing
   database for the files was last closed.

4. A method as in claim 3 wherein the evaluating comprises performing a
   time based comparison without examining a record for each file indexed in the
   indexing database.

5. A machine readable medium providing instructions which when executed
   by a data processing system cause the data processing system to perform a
   method of processing data, the method comprising:
   mounting a storage device;
determining, automatically in response to the mounting, whether to index one or more files on the storage device or to add metadata of the one or more files to a metadata database.

6. A medium as in claim 5 wherein the determining comprises comparing a last time of usage or closing of an indexing database to an unmounting time or last write time of the storage device and also comprises comparing a first time of usage or opening of an indexing database after the unmounting time to a mounting time or first write time of the storage device.

7. A medium as in claim 5 wherein the determining comprises evaluating whether files on the storage device have been modified since an indexing database for the files was last closed.

8. A medium as in claim 7 wherein the evaluating comprises performing a time based comparison without examining a record for each file indexed in the indexing database.

9. A data processing system comprising:
- means for mounting a storage device;
- means for determining, automatically in response to the mounting, whether to index one or more files on the storage device or to add metadata of the one or more files to a metadata database.

10. A system as in claim 9 wherein the determining comprises comparing a last time of usage or closing of an indexing database to an unmounting time or last write time of the storage device and also comprises comparing a first time of usage or opening of an indexing database after the unmounting time to a mounting time or first write time of the storage device.
11. A system as in claim 9 wherein the determining comprises evaluating whether files on the storage device have been modified since an indexing database for the files was last closed.

12. A system as in claim 11 wherein the evaluating comprises performing a time based comparison without examining a record for each file indexed in the indexing database.

13. A machine implemented method of processing data, the method comprising:
   mounting a storage device;
   evaluating, after the mounting, whether to automatically index or update a metadata database for one or more files on the storage device by evaluating whether files on the storage device have been modified or new files have been added since a database for the files on the storage device was last closed or written to.

14. A method as in claim 13 wherein the evaluating comprises performing at least one time based comparison without examining a record for each file indexed or updated in the database.

15. A method as in claim 14 wherein the evaluating further comprises a comparison of write counts.

16. A method as in claim 13 wherein the storage device is a removable storage device which is portable among systems.
17. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
   mounting a storage device;
   evaluating, after the mounting, whether to automatically index or update a metadata database for one or more files on the storage device by evaluating whether files on the storage device have been modified or new files have been added since a database for the files on the storage device was last closed or written to.

18. A medium as in claim 17 wherein the evaluating comprises performing at least one time based comparison without examining a record for each file indexed or updated in the database.

19. A medium as in claim 18 wherein the evaluating further comprises a comparison of write counts.

20. A medium as in claim 17 wherein the storage device is a removable storage device which is portable among systems.

21. A data processing system comprising:
    means for mounting a storage device;
    means for evaluating, after the mounting, whether to automatically index or update a metadata database for one or more files on the storage device by evaluating whether files on the storage device have been modified or new files have been added since a database for the files on the storage device was last closed or written to.
22. A system as in claim 21 wherein the evaluating comprises performing at least one time based comparison without examining a record for each file indexed or updated in the database.

23. A system as in claim 22 wherein the evaluating further comprises a comparison of write counts.

24. A system as in claim 21 wherein the storage device is a removable storage device which is portable among systems.

25. A machine implemented method of processing data, the method comprising:
   mounting a storage device;
   determining, automatically in response to the mounting, whether to index or update a metadata database for one or more files on the storage device without examining a record for each file indexed in the indexing database or having metadata in the metadata database.

26. A method as in claim 25 wherein the determining comprises performing at least one time based comparison of operations in an unmounting prior to the mounting and operations concurrent with or after the mounting.

27. A method as in claim 25 wherein the determining comprises a comparison of write counts.

28. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
   mounting a storage device;
determining, automatically in response to the mounting, whether to index
or update a metadata database for one or more files on the storage
device without examining a record for each file indexed in the
indexing database or having metadata in the metadata database.

29. A medium as in claim 28 wherein the determining comprises performing
at least one time based comparison of operations in an unmounting prior to the
mounting and operations concurrent with or after the mounting.

30. A medium as in claim 28 wherein the determining comprises a
comparison of write counts.

31. A data processing comprising:
means for mounting a storage device;
means for determining, automatically in response to the mounting,
whether to index or update a metadata database for one or more
files on the storage device without examining a record for each file
indexed in the indexing database or having metadata in the
metadata database.

32. A system as in claim 31 wherein the determining comprises performing at
least one time based comparison of operations in an unmounting prior to the
mounting and operations concurrent with or after the mounting.

33. A system as in claim 31 wherein the determining comprises a comparison
of write counts.

34. A machine implemented method of processing data, the method
comprising:
receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or that a new file has been created on the data storage device;
performing an indexing operation on the existing file or the new file in response to the notification, wherein the notification comprises an identifier that identifies the existing file or the new file.

35. A method as in claim 34 further comprising:
generating, by an operating system, the notification.

36. A method as in claim 34 wherein the notification causes an entry, representing the existing file or the new file, to be added to a queue for indexing operations and the queue is saved in a nonvolatile storage.

37. A method as in claim 36 wherein changes to the queue are entered into a log and the log, with the changes, is saved onto the nonvolatile storage before the changes to the queue are saved onto the nonvolatile storage.

38. A method as in claim 34 wherein the indexing operation is stalled or delayed in response to detecting user activity.

39. A method as in claim 34 wherein a processing priority of the indexing operation is reduced in response to detecting user activity.

40. A method as in claim 35 wherein the operating system comprises a kernel for managing processing threads or tasks, a graphics generation component for generating graphics, a windowing component for displaying windows, a file management system for maintaining a directory of files stored on the data storage device.
41. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
   receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or that a new file has been created on the data storage device;
   performing an indexing operation on the existing file or the new file in response to the notification, wherein the notification comprises an identifier that identifies the existing file or the new file.

42. A medium as in claim 41, the method further comprising:
   generating, by an operating system, the notification.

43. A medium as in claim 41 wherein the notification causes an entry, representing the existing file or the new file, to be added to a queue for indexing operations and the queue is saved in a nonvolatile storage.

44. A medium as in claim 43 wherein changes to the queue are entered into a log and the log, with the changes, is saved onto the nonvolatile storage before the changes to the queue are saved onto the nonvolatile storage.

45. A medium as in claim 41 wherein the indexing operation is stalled or delayed in response to detecting user activity.

46. A medium as in claim 41 a processing priority of the indexing operation is reduced in response to detecting user activity.
47. A medium as in claim 42 wherein the operating system comprises a kernel for managing processing threads or tasks, a graphics generation component for generating graphics, a windowing component for displaying windows, a file management system for maintaining a directory of files stored on the data storage device.

48. A data processing system comprising:
   means for receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or that a new file has been created on the data storage device;
   means for performing an indexing operation on the existing file or the new file in response to the notification, wherein the notification comprises an identifier that identifies the existing file or the new file.

49. A system as in claim 48 further comprising:
   means for generating, by an operating system, the notification.

50. A system as in claim 48 wherein the notification causes an entry, representing the existing file or the new file, to be added to a queue for indexing operations and the queue is saved in a nonvolatile storage.

51. A system as in claim 50 wherein changes to the queue are entered into a log and the log, with the changes, is saved onto the nonvolatile storage before the changes to the queue are saved onto the nonvolatile storage.

52. A system as in claim 48 wherein the indexing operation is stalled or delayed in response to detecting user activity.
53. A system as in claim 48 wherein a processing priority of the indexing operation is reduced in response to detecting user activity.

54. A system as in claim 49 wherein the operating system comprises a kernel for managing processing threads or tasks, a graphics generation component for displaying windows, a file management system for maintaining a directory of files stored on the data storage device.

55. A machine implemented method of processing data, the method comprising:
   receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or that a new file has been created on the data storage device;
   performing an indexing operation on the existing file or new file in response to the notification, wherein the notification is not based solely on time or a user input.

56. A method as in claim 55 further comprising:
   generating, by an operating system, the notification.

57. A method as in claim 56 wherein the notification causes an entry, representing the existing file, to be added to a queue for indexing operations and the queue is saved in a nonvolatile storage.

58. A method as in claim 57 wherein changes to the queue are entered into a log and the log, with the changes, is saved onto the nonvolatile storage before the changes to the queue are saved onto the nonvolatile storage.
59. A method as in claim 55 wherein the indexing operation is stalled or delayed in response to detecting user activity.

60. A method as in claim 57 wherein the operating system comprises a kernel for managing processing threads or tasks, a graphics generation component for generating graphics, a windowing component for displaying windows, a file management system for maintaining a directory of files stored on the data storage device.

61. A method as in claim 55 wherein a processing priority of the indexing operation is reduced in response to detecting user activity.

62. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
   receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or that a new file has been created on the data storage device;
   performing an indexing operation on the existing file or new file in response to the notification, wherein the notification is not based solely on time or a user input.

63. A medium as in claim 62, the method further comprising:
   generating, by an operating system, the notification.

64. A medium as in claim 63 wherein the notification causes an entry, representing the existing file or new file, to be added to a queue for indexing operations and the queue is saved in a nonvolatile storage.
65. A medium as in claim 64 wherein changes to the queue are entered into a log and the log, with the changes, is saved onto the nonvolatile storage before the changes to the queue are saved onto the nonvolatile storage.

66. A medium as in claim 62 wherein the indexing operation is stalled or delayed in response to detecting user activity.

67. A medium as in claim 64 wherein the operating system comprises a kernel for managing processing threads or tasks, a graphics generation component for generating graphics, a windowing component for displaying windows, a file management system for maintaining a directory of files stored on the data storage device.

68. A medium as in claim 62 wherein a processing priority of the indexing operation is reduced in response to detecting user activity.

69. A data processing system comprising:
   means for receiving, by an indexing software component, a notification that an existing file on a data storage device has been modified or that a new file has been created on the data storage device;
   means for performing an indexing operation on the existing file or new file in response to the notification, wherein the notification is not based solely on time or a user input.

70. A system as in claim 69 further comprising:
   means for generating, by an operating system, the notification.
71. A system as in claim 70 wherein the notification causes an entry, representing the existing file or new file, to be added to a queue for indexing operations and the queue is saved in a nonvolatile storage.

72. A system as in claim 71 wherein changes to the queue are entered into a log and the log, with the changes, is saved onto the nonvolatile storage before the changes to the queue are saved onto the nonvolatile storage.

73. A system as in claim 69 wherein the indexing operation is stalled or delayed in response to detecting user activity.

74. A system as in claim 71 wherein the operating system comprises a kernel for managing processing threads or tasks, a graphics generation component for generating graphics, a windowing component for displaying windows, a file management system for maintaining a directory of files stored on the data storage device.

75. A system as in claim 69 wherein a processing priority of the indexing operation is reduced in response to detecting user activity.

76. A machine implemented method of processing data, the method comprising:
   - determining a file is to be indexed into an indexing database;
   - adding an entry representing the file into a list for indexing operations;
   - saving the list to a nonvolatile storage.

77. A machine implemented method as in claim 76 further comprising:
removing effectively the entry from the list after indexing the file to create an updated list;
saving the updated list to the nonvolatile storage.

78. A method as in claim 77 wherein the determining comprises receiving a notification that the file is to be indexed, the notification being from an operating system component to an indexing component.

79. A method as in claim 78 wherein the notification is not based solely on time or a user input and a transaction log shows changes to the list.

80. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
determining a file is to be indexed into an indexing database;
adding an entry representing the file into a list for indexing operations;
saving the list to a nonvolatile storage.

81. A medium as in claim 80 the method further comprising:
removing effectively the entry from the list after indexing the file to create an updated list;
saving the updated list to the nonvolatile storage.

82. A medium as in claim 81 wherein the determining comprises receiving a notification that the file is to be indexed, the notification being from an operating system component to an indexing component.

83. A medium as in claim 82 wherein the notification is not based solely on time or a user input and a transaction log shows changes to the list.
84. A data processing system comprising:
means for determining a file is to be indexed into an indexing database;
means for adding an entry representing the file into a list for indexing
operations;
means for saving the list to a nonvolatile storage.

85. A system as in claim 84 further comprising:
means for removing effectively the entry from the list after indexing the
file to create an updated list;
means for saving the updated list to the nonvolatile storage.

86. A system as in claim 85 wherein the determining comprises receiving a
notification being from an operating system component to an indexing
component.

87. A system as in claim 86 wherein the notification is not based solely on
time or a user input and a transaction log shows changes to the list.

88. A machine implemented method of processing data, the method
comprising:
monitoring a user's use of a data processing system;
adjusting automatically, in response to the monitoring, indexing
operations.

89. A method as in claim 88 wherein the indexing operations are performed
automatically by the data processing system in response to notifications, from an
operating system component, to an indexing component that a file needs to be
indexed or re-indexed.
90. A method as in claim 88 wherein as the user's use of the data processing system increases, fewer indexing operations are performed over a period of time.

91. A method as in claim 88 wherein the monitoring comprises determining whether an application program is using all available display area on a display device.

92. A method as in claim 88 wherein the adjusting comprises changing a processing priority of indexing software relative to other software on a data processing system.

93. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:

   monitoring a user's use of a data processing system;

   adjusting automatically, in response to the monitoring, indexing operations.

94. A medium as in claim 93 wherein the indexing operations are performed automatically by the data processing system in response to notifications, from an operating system component, to an indexing component that a file needs to be indexed or re-indexed.

95. A medium as in claim 93 wherein as the user's use of the data processing system increases, fewer indexing operations are performed over a period of time.
96. A medium as in claim 93 wherein the monitoring comprises determining whether an application program is using all available display area on a display device.

97. A medium as in claim 93 wherein the adjusting comprises changing a processing priority of indexing software relative to other software on a data processing system.

98. A data processing system comprising:
   means for monitoring a user’s use of a data processing system;
   means for adjusting automatically, in response to the monitoring, indexing operations.

99. A system as in claim 98 wherein the indexing operations are performed automatically by the data processing system in response to notifications, from an operating system component, to an indexing component that a file needs to be indexed or re-indexed.

100. A system as in claim 98 wherein as the user’s use of the data processing system increases, fewer indexing operations are performed over a period of time.

101. A system as in claim 98 wherein the monitoring comprises determining whether an application program is using all available display area on a display device.

102. A system as in claim 98 wherein the adjusting comprises changing a processing priority of indexing software relative to other software on a data processing system.
103. A machine implemented method of processing data, the method comprising:

determining a time when an indexing database was last updated, the indexing database containing content from files stored on a storage device;
determining whether files stored on the storage device have been modified or created after the time when the indexing database was last updated;
updating the indexing database for any file which was modified or created after the time.

104. A method as in claim 103 wherein the updating occurs automatically without user interaction.

105. A method as in claim 104 wherein the determining whether files stored on the storage device have been modified or created after the time further comprises determining whether the files have been modified or created after a predefined period of time before the time.

106. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:

determining a time when an indexing database was last updated, the indexing database containing content from files stored on a storage device;
determining whether files stored on the storage device have been modified or created after the time when the indexing database was last updated;
updating the indexing database for any file which was modified or created after the time.

107. A medium as in claim 106 wherein the updating occurs automatically without user interaction.

108. A medium as in claim 107 wherein the determining whether files stored on the storage device have been modified or created after the time further comprises determining whether the files have been modified or created after a predefined period of time before the time.

109. A data processing system comprising:

   means for determining a time when an indexing database was last updated, the indexing database containing content from files stored on a storage device;

   means for determining whether files stored on the storage device have been modified or created after the time when the indexing database was last updated;

   means for updating the indexing database for any file which was modified or created after the time.

110. A system as in claim 109 wherein the updating occurs automatically without user interaction.

111. A system as in claim 110 wherein the determining whether files stored on the storage device have been modified or created after the time further comprises determining whether the files have been modified or created after a predefined period of time before the time.
112. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
   determining a file is to have metadata for the file to be added into a metadata database;
   adding an entry representing the file into a list for metadata processing operations;
   saving the list to a nonvolatile storage.

113. A medium as in claim 112 the method further comprising:
   removing effectively the entry from the list after adding metadata for the file to the metadata database to create an updated list;
   saving the updated list to the nonvolatile storage.

114. A medium as in claim 113 wherein the determining comprises receiving a notification that the file is new or modified, the notification being from an operating system component to a metadata software component.

115. A medium as in claim 114 wherein the notification is not based solely on time or a user input and a transaction log shows changes to the list.

116. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:
   monitoring a user's use of a data processing system;
   adjusting automatically, in response to the monitoring, metadata processing operations.
117. A medium as in claim 116 wherein the processing operations are performed automatically by the data processing system in response to notifications, from an operating system component, to a metadata software component that a file needs to have its metadata added to a metadata database.

118. A medium as in claim 116 wherein as the user's use of the data processing system increases, fewer metadata processing operations are performed over a period of time.

119. A medium as in claim 116 wherein the monitoring comprises determining whether an application program is using all available display area on a display device.

120. A medium as in claim 116 wherein the adjusting comprises changing a processing priority of metadata processing software relative to other software on a data processing system.

121. A machine readable medium providing instructions which when executed by a data processing system cause the data processing system to perform a method of processing data, the method comprising:

    determining a time when a metadata database was last updated, the metadata database containing metadata for files stored on a storage device;
    determining whether files stored on the storage device have been modified or created after the time when the metadata database was last updated;
    updating the metadata database for any file which was modified or created after the time.
122. A medium as in claim 121 wherein the updating occurs automatically without user interaction.

123. A medium as in claim 122 wherein the determining whether files stored on the storage device have been modified or created after the time further comprises determining whether the files have been modified or created after a predefined period of time before the time.
FIG. 1
Capture metadata from a variety of different application programs

Make captured metadata available to a searching facility (e.g., a file management system software for searching)

Allow searching of metadata across all applications having captured metadata

Provide User Interface of search engine and search results

FIG. 2
FIG. 3A

JPEG Image File Metadata

ISO Setting               Image Width
Flash Setting             Image Height
f/ Stop                   Image Color Space
Camera Brand Information  Bits/Pixel
User Added Keywords       Other Fields...

FIG. 3B

MP3 Music File Metadata

Artist                   Song Play Times
Genre                    Other Fields...
Album                    
Song names
FIG. 4
Receive notification of change for a file (e.g., receive notification from OS kernel or software application of creation of new file or modification of existing file or deletion of existing file).

Determine type of file (e.g., an Acrobat PDF file or an RTF word processing file or a JPEG image file, etc.)

Activate capture software for determined file type (e.g., a plug in for the application which creates the type of file)

Import appropriate metadata (for particular file type) into a metadata database

Store metadata in a database (e.g., a flat file format database)

Receive search parameter inputs and perform search of metadata database and display results

FIG. 5
FIG. 6
<table>
<thead>
<tr>
<th>Name</th>
<th>Parent</th>
<th>Date Modified</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>today</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FindBrowse</td>
<td>Finder</td>
<td>2/11/04</td>
<td>Folder</td>
</tr>
<tr>
<td>findBrowse sequence2</td>
<td>FindBrowse</td>
<td>2/4/04</td>
<td>Macromedia Director Movie</td>
</tr>
<tr>
<td>yesterday</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>findBrowse multi 3.dir</td>
<td>FindBrowse</td>
<td>2/4/04</td>
<td>Macromedia Director Movie</td>
</tr>
<tr>
<td>FindBrowse.dir</td>
<td>Find Starling Point</td>
<td>2/2/04</td>
<td>Macromedia Director Movie</td>
</tr>
<tr>
<td>findBrowse.sequence</td>
<td>FindBrowse</td>
<td>2/2/04</td>
<td>Macromedia Director Movie</td>
</tr>
<tr>
<td>browse by date5 header5 copy.tif</td>
<td>assets</td>
<td>2/2/04</td>
<td>Adobe Photoshop TIFF file</td>
</tr>
<tr>
<td>findBrowse seq assets</td>
<td>FindBrowse</td>
<td>2/2/04</td>
<td>Folder</td>
</tr>
<tr>
<td>before yesterday</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>findBrowse assets</td>
<td>FindBrowse</td>
<td>2/2/04</td>
<td>Folder</td>
</tr>
<tr>
<td>find Browse multi 2.dir</td>
<td>FindBrowse</td>
<td>2/2/04</td>
<td>Macromedia Director Movie</td>
</tr>
<tr>
<td>find Browse multi 1.dir</td>
<td>FindBrowse</td>
<td>2/2/04</td>
<td>Macromedia Director Movie</td>
</tr>
<tr>
<td>browse by keyword map + SB2.psd</td>
<td>Find</td>
<td>1/12/04</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by keyword map + SB.psd</td>
<td>Find</td>
<td>1/9/04</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>over a week ago</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>browse by keyword map.psd</td>
<td>Finder</td>
<td>12/17/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 + info2.psd</td>
<td>Finder</td>
<td>12/15/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by folder + info.psd</td>
<td>Finder</td>
<td>12/15/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 + info.psd</td>
<td>Finder</td>
<td>12/15/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 + actions2.psd</td>
<td>Finder</td>
<td>12/12/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 + actions.psd</td>
<td>Finder</td>
<td>12/12/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 header5.psd</td>
<td>Finder</td>
<td>12/11/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 header4.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 header3.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 header2.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5 header.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by date5.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by day4.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by day3.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by day2.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
<tr>
<td>browse by day.psd</td>
<td>Finder</td>
<td>12/10/03</td>
<td>Adobe Photoshop file</td>
</tr>
</tbody>
</table>
29/53

![Smart Folder Diagram](image)

- **Icon size:** 128 x 128
  - Small
  - Large
- **Text size:** 12 pt
- **Label position:**
  - Bottom
  - Right
- **Options:**
  - Snap to grid
  - Show item info
  - Show icon preview
  - Keep arranged by
    - Name
- **Background:**
  - White
  - Color
  - Picture

**FIG. 14D**
FIG. 15A
**Search: button**

**Documents**
- button.psd  
  - September 19 3:55 pm
- Action Button.dit  
  - June 2003
- Action Button.dit  
  - June 2003
- Metal buttons.psd  
  - March 2003
- anim.gif.x  
  - December 2002

**Folders**
- actionbutton  
  - January 5 4:45 pm
- new buttons  
  - April 2003
- action button inc  
  - February 2003
- alex - clear button  
  - December 2002
- Help Button  
  - November 2002

**HTML**
- 10Buttons8.html  
  - November 1999
- 10Buttons9.html  
  - November 1999
- 10Buttons4.html  
  - November 1999
- 10Buttons8.html  
  - November 1999
- 10Buttons9.html  
  - November 1999

**Images**
- button oil.gif  
  - November 2002
- button ace.gif  
  - November 2002
- button nd.gif  
  - November 2002
- buttonlayer.gif  
  - November 2002
- PushButton_to.png  
  - November 2002

- PushButton.png  
  - November 2002
- 12_01..button  
  - November 2002
- 12_02..button  
  - November 2002
- 12_03..button  
  - November 2002
- 12_04..button  
  - November 2002

- 12_05..button  
  - November 2002
- 12_18..button  
  - November 2002
- 12_19..button  
  - November 2002
- 12_22..button  
  - November 2002
- 13_5..button  
  - November 2002

- 13_6..ulction  
  - November 2002
- 14_00..Button  
  - November 2002
- 16_00..ulction  
  - November 2002
- 16_1..editor  
  - November 2002
- 16_2..Btn  
  - November 2002

**Group By:**
- Kind
- Date
- People
- Flat List

**Within Group Sort by:**
- Name
- Date
- Kind
- People

**Search:**
- Everywhere
- Home
- Text (work)
- Scratch
- hd

**FIG. 18B**
<table>
<thead>
<tr>
<th>Search: Imran</th>
<th>30 Results</th>
<th>Group By: Kind</th>
<th>People</th>
<th>Flat List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Imran Chaudhri</strong></td>
<td>June 2000</td>
<td>Date</td>
<td>Name</td>
<td>Home</td>
</tr>
<tr>
<td>CD Player Bits</td>
<td>June 2000</td>
<td></td>
<td></td>
<td>text</td>
</tr>
<tr>
<td>Required Changes to the AAP (B.</td>
<td>May 2000</td>
<td></td>
<td></td>
<td>work</td>
</tr>
<tr>
<td>Icon Text</td>
<td></td>
<td></td>
<td></td>
<td>scratch</td>
</tr>
</tbody>
</table>

| **No Authors** | | | | |
| IMAP-imran@mail.apple.com | Thursday 4:38 pm | | | Every |
| Drafts (imran@apple.com).mbox | June 17 2:06 am | | | where |
| Imran | June 15 2:16 pm | | | text |
| Sent Messages (imran@apple.com).mbox | June 10 9:23 pm | | | notes |

| **Sarah A. Brody** | | | | |
| pl_agenda_082201.doc | November 2002 | | | scr |
| pl_minutes_081701.doc | November 2002 | | | h |
| pl_timeline_082201.doc | November 2002 | | | 
| minutes_073101.doc | November 2002 | | | 

| **Susan McGarry** | | | | |
| WMDC 2000 - Hit Labs | May 2000 | | | |

**FIG. 19A**
<table>
<thead>
<tr>
<th>Search: imran</th>
<th>30 Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD Player bin</td>
<td>June 2000</td>
</tr>
<tr>
<td>Required Changes to the AAF (B)</td>
<td>June 2000</td>
</tr>
<tr>
<td>Icon Text</td>
<td></td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
</tr>
<tr>
<td>Copyright</td>
<td></td>
</tr>
<tr>
<td>Last used date</td>
<td>May 2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Name</th>
<th>Creation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl_agenda_082201.doc</td>
<td>November 2002</td>
</tr>
<tr>
<td>pl_minutes_081701.doc</td>
<td>November 2002</td>
</tr>
<tr>
<td>pl_timeline_082201.doc</td>
<td>November 2002</td>
</tr>
<tr>
<td>minutes_073101.doc</td>
<td>November 2002</td>
</tr>
<tr>
<td>WWDC 2000 - Hi Late</td>
<td>May 2000</td>
</tr>
<tr>
<td>imran</td>
<td>May 2003</td>
</tr>
<tr>
<td>window pic for imran</td>
<td>July 2 4:36 pm</td>
</tr>
<tr>
<td>window pic for imran</td>
<td>July 7 11:42 am</td>
</tr>
<tr>
<td>imran</td>
<td>Thursday 4:39 pm</td>
</tr>
<tr>
<td>Help for imran</td>
<td>January 2000</td>
</tr>
<tr>
<td>system_preferences_v_imran.psd</td>
<td>January 2001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Name</th>
<th>Creation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>nhl100.psd</td>
<td>December 1999</td>
</tr>
<tr>
<td>RID-IMRAN</td>
<td>October 2001</td>
</tr>
<tr>
<td>140 - Aqua Overview 5/14 -imran</td>
<td>May 2000</td>
</tr>
<tr>
<td>140 - Aqua Overview 5/14 -imran</td>
<td>May 2000</td>
</tr>
<tr>
<td>140 - Aqua Overview 5/14 -imran</td>
<td>May 2000</td>
</tr>
<tr>
<td>140 - Aqua Overview 5/14 -imran</td>
<td>May 2000</td>
</tr>
<tr>
<td>Intranet</td>
<td>September 5 1:21 pm</td>
</tr>
<tr>
<td><a href="mailto:IMAP-imran@mail.apple.com">IMAP-imran@mail.apple.com</a></td>
<td>Thursday 3:50 pm</td>
</tr>
<tr>
<td>Junk <a href="mailto:imran@apple.com.mbox">imran@apple.com.mbox</a></td>
<td>May 24 5:29 pm</td>
</tr>
<tr>
<td>Sent Messages (<a href="mailto:imran@apple.com.mbox">imran@apple.com.mbox</a>)</td>
<td>June 10 5:23 pm</td>
</tr>
<tr>
<td>Drafts (<a href="mailto:imran@apple.com">imran@apple.com</a>).mbox</td>
<td>June 17 2:06 am</td>
</tr>
<tr>
<td>Southwest Airlines No Tickets Required Confirmation</td>
<td>April 29 11:45 am</td>
</tr>
<tr>
<td>LBT</td>
<td>May 24 1:21 pm</td>
</tr>
<tr>
<td>imran</td>
<td>June 15 2:16 pm</td>
</tr>
</tbody>
</table>

**FIG. 19B**
### FIG. 19D

<table>
<thead>
<tr>
<th>Year</th>
<th>Search: te</th>
<th>84 Results</th>
<th>Group By:</th>
<th>Kind</th>
<th>Date</th>
<th>People</th>
<th>Flat List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1925</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1950</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1903</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1905</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1907</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1908</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1952</td>
<td>Today</td>
<td>Today 11:12 am</td>
<td>Kind</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1954</td>
<td>Yesterday</td>
<td>Yesterday 8:48 pm</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1956</td>
<td>Last week</td>
<td>June 16 11:05 am</td>
<td>People</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Search:
- Type
- Everywhere
- Name
- Home
- todo (work)
- scratch
- hd

#### Last week:
- 15 Vie Gat Tha.m4a
- 14 The Brouchaha.m4a
- 13 Crawlspace.m4a
- 12 An Open Letter To NYC.m4a
- 11 Shazam!.m4a

#### Later:
- Index.html
- web share.gif
- apache_pb.gif
- maoosxlogo.gif

#### Today:
- Picture 23
- info.plist
- ascii indent
- SignatureByAccount.plist
- MessageSorting.plist.backup
- 94 more...
Display a system wide Menu for inputting search queries

2001

Input search, and as search query is inputted, perform the search (begin searching and displaying search results before user finishes inputting search query)

2003

Perform search through files, metadata, emails within an email program, address book entries within an address book, calendar entries within a calendar program, etc.

2005

Display an abbreviated (incomplete) list of hits (if there are more than a certain number of hits), sorted by relevance and segregated into groups (categories)

2007

Receive command to display all hits

2009

Display a search results window with ability to display 2 different types of view (e.g., icon and list views) within the same closeable, resizable, moveable window

2011

FIG. 20
Receive one or more inputs indicating a selection of a plurality of items (e.g., data files, folders, application programs, etc.)

Receive a command requesting both the creation of a new storage facility and an association of the plurality of items with the new storage facility

Create the new storage facility (e.g., a new folder with a predetermined directory pathname or a user specified directory pathname) and associate (e.g., move or copy) the selected plurality of items with the new storage facility

FIG. 21
Figure 23

Monitor user's use of data processing system

If user is using the system beyond a threshold level or if the system is in a game mode (e.g., the full screen is being used by a program), then either stop indexing and/or importing metadata or set the indexing and/or metadata software to a lower priority (e.g., set processing priority to lowest priority for both the indexing software and the metadata software).

If user is using the system below a threshold level and the system is not in a game mode, then give indexing and/or metadata software a higher processing priority.
Figure 24

Generate a notification automatically in response to saving changes to an existing file or saving a newly created file (e.g., an operating system's kernel automatically generates a notification in response a user or system initiated save (e.g., writing the file to a hard drive)).

Indexing software receives the notification and optionally enters the notification into a "to be indexed" queue (note: the notification is not based solely on time).

Indexing software performs indexing of the file and stores indexing data into an index database.
Figure 25

2501
Receive notification from operating system kernel that an existing file has changed or a new file has been created

2503
Enter an identifier for the file into an indexing queue and store the queue in a non-volatile storage (e.g., a hard drive) and enter information about changes to the queue into a transaction log

2505
Perform indexing for files listed in the indexing queue

2507
Remove identifier for file from the indexing queue once the file has been indexed and store updated queue in non-volatile storage and enter information about changes to the queue into the transaction log

2509
Interrupt indexing in response to increased use by user

2511
Resume indexing in response to decreased use by users (resume indexing with files listed in the indexing queue which may act as a FIFO buffer)
Figure 26

2601

Determine when file system's index database was last updated

2603

Determine if any file was updated (or created) after last update time of index database (or after 1 hr. earlier than the last update time of the index database)

2605

Update index database for any file updated or created after last update time of the index database (or after 1 hr. earlier than the last update time of the index database)
Figure 27

2701
Finish indexing and close the index database and index queue and unmount volume which has been indexed (at least partially) by a first data processing system

2703
Use volume on another system (e.g., the volume is removable and is used on another system), possibly modifying files on the volume (without updating the index database)

2705
Mount the volume again on the first data processing system (note: the volume normally stores the index database and index queue)

2707
Determine whether any files on the volume have been modified after the index database was last closed
At shutdown or unmounting of volume:

record $T_{bs} =$ time index database was closed
(may be last write time)

record $T_{fs} =$ time file system (on volume) was
unmounted

record $W_{s} =$ write count for volume at
shutdown or unmounting

At time of mounting of Volume

collect:  $T_{FM} =$ last mount time

$T_{Do} =$ database open time

$W_{0} =$ current write count

Do not update index database if:

$T_{Fs} - T_{bs} < \text{small delay}$

and

$T_{Do} - T_{FM} < \text{small delay}$

and

$W_{0} - W_{s} < \text{small number}$

otherwise, update the index database