SYSTEM AND METHOD UTILIZING VIRTUAL FOLDERS

CROSS-REFERENCE(S) TO RELATED APPLICATION(S)

This application is related to U.S. applications attorney docket Nos. MSFT-1-20699 and MSFT-1-20700, filed concurrently with the present application, which are hereby incorporated by reference in their entireties.

FIELD OF THE INVENTION

The present invention relates to file systems, and more particularly, to a system and method utilizing virtual folders.

BACKGROUND OF THE INVENTION

Present computer file systems have a number of undesirable limitations. One limitation is that users are generally unable to control the structure that they are shown. In other words, when folders are organized, a user must choose a structure, and that structure is then difficult to change. As a specific example, for a "music" folder, a user may choose to organize the music files in an artist/album format, wherein all of the album folders for each artist are grouped into that particular artist's folder, and all of the songs on a particular album are grouped into that album's folder. The artist/album format is not conducive to playing a type of music (e.g., playing two jazz songs from two different artists), or for playing a selection of albums from different artists.

As another issue, a user may have a large number of files which are difficult to organize. Some users implement a rigid sense of placement for the files, and thus create strict hierarchies for them. The management of such files become increasingly complex and difficult as the number of available documents grows, making search and retrieval also difficult. This problem is further exacerbated when additional files are utilized from other locations, such as shared files, etc.

Users also have to deal with files being in different locations, such as on different devices, on other PCs, or online. For example, users can select to listen to their music on the computer (as may be accessible to a music program) or can go online and listen to music from Web sites, however there is a strict division between these two sources. Music coming from different locations is organized differently, and not kept in the same

fashion or place. As another example, files stored on a corporate network may inherently be separated from files a user has on a current machine.

Users also have to keep track not only of what file data is stored, but where it is stored. For example, for music files, users are forced to keep copies on various systems and to try to track which music files are located where. This can make files difficult to locate, even when they are locally stored.

It is also sometimes difficult to find and return to files that a user has. A user may find it difficult to recall where and how they stored certain files. Given a set of folders and even a group of similar files, users often find it difficult to quickly find the one that they are looking for. For files stored in a difficult place to find, it is that much more complex to locate. In addition, once users have enough files in a folder, it becomes more difficult to parse the folder quickly, especially if the contents are similar.

It is also sometimes difficult for users to find or return to files on a network. Sharing and publishing files is often hard to do, and it may often be even more difficult to retrieve such a file from someone who makes it available. Users typically have to memorize or map the various sites and names that they need for finding files on a network.

Name spaces may vary, which can cause confusion to the user as to what is "correct." This is particularly true on a network where there are different naming conventions, limitations, and so on. For example, certain operating systems may require short names with no spaces in order for them to be visible.

Programs also often save files to their own directory or other name spaces, which can make it difficult for users to find their way back to the files. Programs often have default directories and places they save documents. A user often has to search through their hard disk and make guesses about where a file is stored.

Related items are also often stored in separate places. Related files that a user has may be stored on different parts of the hard disk, etc. This problem becomes more common with the developments of digital media services that have multiple content types (e.g., pictures, music, video).

The present invention is directed to providing a system and method that overcome the foregoing and other disadvantages. More specifically, the present invention is directed to a file system utilizing virtual folders.

SUMMARY OF THE INVENTION

A system and method utilizing virtual folders is provided. In accordance with one aspect of the invention, the virtual folders expose regular files and folders (also known as directories) to users in different views based on their metadata instead of the actual physical underlying file system structure on the disk. Thus, the system is able to take a property that is stored in the database and represent it as a container that is like a folder. Since users are already familiar with working with folders, by presenting the virtual folders in a similar manner, users can adapt to the new system more quickly.

In accordance with another aspect of the invention, the virtual folders are provided according to a method that is utilized in a computer system having a display and a memory for storing the items. In accordance with the method, a metadata property is selected. The system then searches for items that have the selected metadata property, and a virtual folder display object is provided that represents the collection of items that have the metadata property.

In accordance with another aspect of the invention, the system includes a folder processor that obtains queries from a user and a relational database for storing information about the items. The folder processor first obtains a query from a user and passes the query to the relational database. The relational database provides results back to the folder processor, and based on the results from the relational database, the folder processor provides the results to the user as virtual folders. In one embodiment, the results that are provided back to the folder processor include database rows and columns. The database rows and columns are converted by the folder processor into an enumerator structure, which is then used to populate the display with the resulting virtual folders.

In accordance with another aspect of the invention, users are able to work with the virtual folders through direct manipulation. In other words, the mechanisms that are provided for manipulating the virtual folders are similar to those that are currently used for manipulating conventional physical folders (e.g., clicking and dragging, copying, pasting, etc.).

In accordance with another aspect of the invention, the method for performing the direct manipulation of the virtual folders is provided in a computer system having a display and a memory for storing the items. In accordance with the method, groups of items are represented as virtual folders. Defined actions are provided that can be performed for direct manipulation of the virtual folders, wherein when a defined action is

performed, the virtual folder is manipulated as directed by the defined action. An example of a defined action would be clicking and dragging a virtual folder. In one embodiment, the action of clicking and dragging a first virtual folder to a second virtual folder performs the function of copying the items from the first virtual folder to the second virtual folder. The copying of items to a virtual folder may involve adding or otherwise altering selected metadata properties that are associated with the items.

In accordance with another aspect of the invention, filters are provided for manipulating the virtual folders. The filters are essentially tools for narrowing down a set of items. In one embodiment, the filters are dynamically generated based on the properties of the separate items. For example, for a set of items, the filter mechanism may review the properties, and if the items generally have "authors" as a property, the filter can provide a list of the authors. Then by clicking on a particular author, the items that don't have the author disappear. This allows the user to narrow the contents.

In accordance with another aspect of the invention, quick links are provided. In one embodiment, quick links are a set of predefined links (e.g., located on the left side of the display) that can be clicked on to generate useful views of the sets of items. These can be predefined by the program, or set by a user. For example, clicking on "all authors" could return a view stacked by authors. "All documents" may return a flat view of all the documents across all of the storage areas. Users can also create their own quick links. For example, a user might filter down to all of the documents that they modified in January 2003, and then could save that as a quick link.

In accordance with another aspect of the invention, libraries are provided. Libraries consist of large groups of usable types of files that can be associated together. For example, photos may be one library, music may be another, and documents may be another. The libraries provide tools and activities that are related to the particular types of items. For example, in the photo library, there are tools and filters that relate to manipulating photos, such as for creating slide shows or sharing pictures.

In accordance with another aspect of the invention, a wide scope of files or items may be available. In other words, the system is able to represent files/items from multiple physical locations (e.g., different hard drives, different computers, different network locations, etc.) so that to a user all the items appear to be from one location. For example, a user can be presented with all of their music files on a single screen, and

manipulate the files all from one view, even though the files may be physically stored on different hard drives, different computers, or different network locations.

In accordance with another aspect of the invention, non-file items may be represented in the virtual folders. In other words, files that are stored in memory are located in a physical store. The virtual folders can be made to include items that are not currently represented in the physical store. Examples of non-file items are e-mails, and contacts.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIGURE 1 is a block diagram of a general purpose computer system suitable for implementing the present invention;

FIGURE 2 is a block diagram of a virtual folder system in accordance with the present invention;

FIGURE 3 is a flow diagram illustrative of a routine by which a user provides a query that draws back selected files and folders;

FIGURE 4 is a flow diagram illustrative of a routine by which virtual folders are constructed and displayed on the screen in accordance with either a default query or a query from the user;

FIGURE 5 is a tree diagram of a folder structure in accordance with a physical folder arrangement on a hard drive;

FIGURE 6 is a tree diagram of a virtual folder structure;

FIGURE 7 is a tree diagram of the virtual folder structure of FIGURE 6, wherein the clients stack is further filtered by contracts and year;

FIGURE 8 is a tree diagram of the virtual folder structure of FIGURE 7, wherein the contracts of the clients stack are further filtered by year;

FIGURE 9 is a tree diagram of the virtual folder structure of FIGURE 6, wherein the contracts stack is further filtered by clients and year, of which the clients are still further filtered by year;

FIGURE 10 is a diagram illustrative of a screen display showing the stacks of a document library;

FIGURE 11 is a diagram illustrative of a screen display showing the documents in the ABC Corp. stack of FIGURE 10;

FIGURE 12 is a diagram illustrative of a screen display in which a stacking function is selected for the documents of FIGURE 11;

FIGURE 13 is a diagram illustrative of a screen display in which a "stack by author" parameter is selected for the stacking function of FIGURE 12;

FIGURE 14 is a diagram illustrative of a screen display in which the files of FIGURE 13 have been stacked by author;

FIGURE 15 is a diagram illustrative of a screen display in which a stacking function is selected and a "stack by category" option is further selected for restacking the files of FIGURE 14;

FIGURE 16 is a diagram illustrative of a screen display in which the files of FIGURE 14 have been restacked by category;

FIGURE 17 is a diagram illustrative of a screen display in which a quick link for showing physical folders is selected;

FIGURE 18 is a diagram illustrative of a screen display in which the physical folders are shown which contain the files of the virtual folder stacks of FIGURE 17;

FIGURE 19 is a flow diagram illustrative of a routine by which a user can directly manipulate virtual folders;

FIGURE 20 is a diagram illustrative of a screen display in which a new "West Coast" stack has been added to the stacks of FIGURE 10;

FIGURE 21 is a diagram illustrative of a screen display in which direct manipulation is used for copying the files from the "ABC Corp." stack to the "West Coast" stack of FIGURE 20;

FIGURE 22 is a flow diagram illustrative of a routine for the system dynamically generating new filter terms;

FIGURE 23 is a flow diagram illustrative of a routine for the system filtering items based on the selection of a filter term;

FIGURE 24 is a diagram illustrative of a screen display in which the stacks of FIGURE 10 have been filtered by the term "AB";

FIGURE 25 is a diagram illustrative of a screen display in which the stacks of FIGURE 10 have been filtered by the term "ABC";

FIGURE 26 is a diagram illustrative of a screen display in which the filter term "year 2002" is selected for the stacks of FIGURE 10;

FIGURE 27 is a diagram illustrative of a screen display in which the stacks of FIGURE 10 have been filtered by the "year 2002" and the further selection of the filter term "month";

FIGURE 28 is a diagram illustrative of a screen display in which a list is presented for selecting a month for filtering;

FIGURE 29 is a diagram illustrative of a screen display wherein the stacks of FIGURE 10 have been further filtered by the month of January, and further showing a filter term of "day";

FIGURE 30 is a flow diagram illustrative of a routine for creating a new quick link;

FIGURE 31 is a diagram illustrative of a screen display for creating a new quick link called "January Work" based on the filtering of FIGURE 29;

FIGURE 32 is a diagram illustrative of a screen display in which a quick link of "All Authors" is selected;

FIGURE 33 is a diagram illustrative of a screen display in which a list of all of the authors of FIGURE 32 is presented;

FIGURE 34 is a diagram illustrative of a screen display in which "Author 1" has been selected from the list of FIGURE 33 and all of the Author 1's documents are shown;

FIGURE 35 is a flow diagram illustrative of a routine for creating a new library;

FIGURE 36 is a diagram illustrative of a screen display in which a collection of various available libraries are shown;

FIGURE 37 is a flow diagram illustrative of a routine for defining the scope of a virtual folder collection;

FIGURE 38 is a block diagram illustrative of the various sources which may form the scope of a virtual folder collection;

FIGURE 39 is a flow diagram illustrative of a routine for including non-file items in a virtual folder collection; and

FIGURE 40 is a diagram illustrative of a screen display showing various non-file items included in a virtual folder.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is directed to virtual folders. Virtual folders utilize the same or similar user interfaces that are currently used for file systems. The virtual folders expose regular files and folders (also known as directories) to users in different views based on their metadata instead of the actual physical underlying file system structure on the disk. Location-independent views are created which allow users to manipulate their files and folders utilizing similar controls as those presently used for managing file systems. In general, this means that users can organize and rearrange their files based on inherent properties in the files themselves, instead of the managing and organization being done as a separate part of the system. The virtual folders may represent files or items from different physical locations, such as from multiple disk drives within the same computer, between multiple computers, or different network locations, such that one view of files or items can expose files or items sitting at different physical locations. In one embodiment, the different items or files need only be connected via an IP network in order to be included.

The virtual folder modeling is also able to be used for traditionally non-file entities. An application of this is to have a set of user interfaces similar to files and folders (that is, objects and containers) to show traditionally non-file entities. One example of such non-file entities would be e-mails, while another would be contact information from a contact database. In this manner, virtual folders provide for a location-independent, metadata-based view system that works regardless of whether the data being shown is from files or non-file entities. In general, these aspects allow more flexibility in terms of letting users manipulate their files and data, using both common user interface techniques (drag and drop, double-click, etc.) as well as leveraging the rich integration of various data types.

FIGURE 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the present invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, characters, components, data structures, etc., that perform particular tasks or implement particular abstract data types. As those skilled in the art will appreciate, the invention may be practiced with other computer system configurations, including hand-held

devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIGURE 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20, including a processing unit 21, system memory 22, and a system bus 23 that couples various system components including the system memory 22 to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system (BIOS) 26, containing the basic routines that helps to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27 for reading from or writing to a hard disk 39, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31, such as a CD-ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer-readable instructions, data structures, program modules, and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk 39, a removable magnetic disk 29, and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk 39, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35, one or

more application programs 36, other program modules 37 and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus 23, but may also be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A display in the form of a monitor 47 is also connected to the system bus 23 via an interface, such as a video card or adapter 48. One or more speakers 57 may also be connected to the system bus 23 via an interface, such as an audio adapter 56. In addition to the display and speakers, personal computers typically include other peripheral output devices (not shown), such as printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more personal computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20. The logical connections depicted in FIGURE 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local area network 51 through a network interface or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20 or portions thereof may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary, and other means of establishing a communications link between the computers may be used.

As implemented on a system of the type illustrated in FIGURE 1, the present invention utilizes virtual folders which make it easier for users to perform basic tasks around file manipulation and folder navigation (browsing) and to provide higher level

storage capabilities which can be leveraged in new features. The virtual folders expose files and items to users in different views based on their metadata instead of the actual physical underlying file system structure on the disk.

FIGURE 2 is a block diagram of a virtual folder system 200 in accordance with the present invention. As will be described in more detail below, the virtual folders allow a user to change the "pivot" which controls the way the data is viewed. As an example, a user could view their music as a flat list of all the songs, which can be grouped by album. Alternatively, the user could switch the view to show only the genres or artists or years, etc. The user can tailor the view to see only the objects suited to the task at hand. This allows an improved browsing experience that negates the need for further navigation through folders (both down and back up). The same lessons and capabilities apply to modeling other data-types not stored as files. Contacts, for example, can be exposed to the user in this way, giving them familiar interface capabilities, as well as richer infrastructure for manipulating them than is provided by a flat address book.

As illustrated in FIGURE 2, the virtual folder system 200 includes a folder processor 210, a relational database 230, a virtual folder descriptions database 232, an other shell folders component 234, a folder handler's component 236, and a shell browser and view component 240. The folder processor 210 includes a native handling code component 212, a handler factory component 214, a property writer component 216, a rowset parser component 218, a query builder component 220, an enumerator component 222, and a property factory component 224.

The relational database 230 stores properties about all files in the system. It also stores some items, like contacts (i.e., non-file items), entirely. In general, it stores metadata about the types of files and items that it contains. The relational database 230 receives SQL queries from the query builder 220. The relational database 230 also sends SQL rowsets to the rowset parser component 218, with one row per item column, columns being the item properties.

The virtual folder descriptions database 232 includes the virtual folder descriptions. The virtual folder descriptions database 232 sends data to the query builder component 220, including a list of types to display in the folder, the initial filter, and the physical locations to show results from (the scopes).

With regard to the other shell folders component 234, the folder processor 210 delegates to existing shell folders from many types of items, including all files, for

handlers or properties. The other shell folders component 234 sends properties from other folders to the property factory 224. The other shell folders component also sends handlers to the handler factory 214.

The folder handlers component 236 provides code behavior for the items that exist only in the database, like contacts. This is what allows non-file items to behave akin to files. The folder handlers component 236 sends handlers to the handler factory 214.

For the native handling code component 212, the folder processor 210 directly implements certain handlers based on the properties of the items. The native handling code component 212 sends handlers to the handler factory 214. For the native handling code component 212 and the folder handlers component 236, like all namespaces, virtual folders have to provide a set of handlers (context menu, icon, thumbnail, infotip, . . .) for their items. For most of these (infotip, data object, drag-drop handler, background context menu . . .) the virtual folder provides a common (native) handler for all the types it holds. However there are others which the author of the type has to provide (context menu on the item itself, writable property store, . . .). The default handler can also be overridden. Virtual folders reuse this for files and allow non-file items do the same.

The handler factory 214 takes ID lists and produces code behaviors that provide context menus, icons, etc. In general, the folder processor 210 may use native handlers, external handlers, or delegate to other shell folders to get handlers, as described above with respect to the native handling code component 212, the other shell folders component 234, and the folder handlers component 236. The handler factory component 214 sends handlers to the shell browser in view 240, as requested by the view. The handler factory component 214 sends a property handler to the property writer 216.

The property writer 216 converts user intentions such as cut, copy, and paste into property rights to the file or item. A shell browser and view component 240 sends data to the property writer 216, including direct manipulation (cut/copy/paste) or editing of metadata. In general, since virtual folders present an organization based on the properties of an item, operations such as move and copy (drag-drop) become an edit on those properties. For example, moving a document, in a view stacked by author, from Author 1 to Author 2, means changing the author. The property writer component 216 implements this function.

The rowset parser 218 takes database rowsets and stores all item properties into a shell ID list structure. A rowset takes the piecewise definition of the virtual folder and

builds a SQL string which can then be issued to the database. The rowset parser component 218 sends ID lists to the enumerator component 222. As described above, the rowset parser component 218 also receives data from the relational database 230, including SQL rowsets, with one row per item, the columns being item properties.

The query builder component 220 builds SQL queries. The query builder component 220 receives data from the enumerator component 222, including new filters from the navigation. The query builder component 220 also receives data from the virtual folder descriptions database 232, including a list of the types to display in the folder, the initial filter, and the physical location to show results from (the scopes). The query builder component 220 sends the SQL queries to the relational database 230.

In general, the query builder component 220 includes a set of rows (in other words a table). This is what running the query yields. The rowset parser component 218 takes each row and using the column names transforms the row into an ID list. An ID list is a well-known shell structure which is used to reference items in a namespace. Doing this allows virtual folders to be just like any other namespace to the rest of the shell. Also caching this data helps keep database access, which can be costly, to a minimum.

The enumerator component 222 operates in response to a navigation to a virtual folder. As described above, the enumerator component 222 receives ID lists from the rowset parser component 218, and sends new filters from the navigation to the query builder component 220. The enumerator 222 also sends data to the shell browser and view component 240, including ID lists that are returned to be inserted into the view after a navigation.

The property factory component 224 takes ID lists and property identifiers and returns values for those properties. The property factory component 224 receives data from the handler factory component 214 including the property handler. As described above, the property factory component 224 also receives data from the other shell folders component 234, including properties from other folders. The property factory component 224 also sends data to the shell browser and view component 240, including item properties, as requested by the view.

The shell browser and view component 240 displays the contents of a folder in a window, and handles all the user interaction with the displayed files or items, such as clicking, dragging, and navigating. Thus, the shell browser and view component 240

receives the user actions. The shell browser and view component 240 also gets the data regarding the code behaviors that it needs from the folder, in this case the folder processor 210.

As described above, the virtual folders expose regular files and folders (also known as directories) to users in different views based on their metadata instead of the actual physical underlying file system structure on the disk. Thus, the system is able to take a property that is stored in the database and represent it as a container that is like a folder. Since users are already familiar with working with folders, by presenting the virtual folders in a similar manner, users can adapt to the new system more quickly.

FIGURE 3 is a flow diagram illustrative of a routine 300 by which a user provides a query that draws back selected items. At a block 302, the folder processor gets a query from the user. In a block 304, the folder processor passes the query to the relational database. At a block 306, the relational database provides the results back to the folder processor. At block 308, the folder processor provides the results to the user in the form of virtual folders and items.

FIGURE 4 is a flow diagram illustrative of a routine 320 by which virtual folders are constructed and displayed on the screen in accordance with either a default query or a query from the user. At a block 322, when a user first opens the virtual folder, a default query is used. This default query is taken from the registry. For example, the default query for a music library could be to show all the songs grouped by album. At a block 324, the folder processor constructs a query object for this query, and then passes this query to the relational database. At a block 326, the relational database generates the results of the query and passes these back to the folder processor as database rows and columns.

At a block 328, the folder processor takes these results and converts them from the rows and columns of data into an enumerator structure, which is used by the folder view to populate the screen with the resulting virtual folders and items for the user to interact upon. At a decision block 330, a user decides whether to change the view (by issuing a different query or "pivot"). For example, a user could issue a "show all artists" pivot. If the user does want to change the view, then the routine returns to block 324 where the folder processor passes this new query to the relational database, and receives back new rows and columns of results, and constructs a new enumerator structure. The

process then continues as described above, as the folder view clears and updates, using the enumerator to draw the "artist" objects to the screen.

In one example, album objects are provided that represent containers that users can navigate into. For example, double-clicking the "Beatles" albums will navigate the view to see all of the Beatles' songs. The folder processor issues the "show all Beatles' songs" query to the relational database, which hands back the rows and columns of data for those songs. The folder processor creates an enumerator of all these songs, which then get drawn to the screen.

The user can also choose the view at any point while browsing virtual folders. From the above example, after narrowing down to just show Beatles songs, a user can change the view to only show the songs as albums. The process of changing the view of items into another representation is called "stacking". This is because the items are conceptually arranged into "stacks" based on that representation. In this case, the songs are rearranged into stacks for each of the various albums. Users can then navigate into one of these stacks, only seeing the songs from that particular album. Again, the user can rearrange the view of these remaining songs into stacks based on a property (e.g., a rating, for example). If the rating property were selected, the songs from that Beatles album would be shown in stacks for a one-, two-, or a three-star rating.

The results of each query depend on which physical locations are included in the scope. For example, the scope may be made to include only the folders in the user's "my documents" folder. Alternatively, the scope could include all folders on the computer, or even all folders on multiple network connected computers. The user is able to view and change the scope through a scope property sheet. In one example, the scope property sheet could be exposed by right-clicking on the virtual folder and choosing "properties." The user could add new folders to the scope, or remove folders that were previously added.

One group of users for which virtual folders will provide particular utility is knowledge workers. Virtual folders allow knowledge workers to easily switch between viewing documents by file type, project, case number, author, etc. Since knowledge workers each tend to have a different method for organizing documents, virtual folders can be used to accommodate these different preferences.

FIGURE 5 is a tree diagram of a folder structure in accordance with a physical folder arrangement on a hard drive. This physical folder arrangement is based on the

traditional implementation of folders, which may be based on NTFS or other existing file systems. Such folders are referred to as physical folders because their structuring is based on the actual physical underlying file system structure on the disk. As will be described in more detail below, this is in contrast to virtual folders, which create location-independent views that allow users to manipulate files and folders in ways that are similar to those currently used for manipulating physical folders.

As illustrated in FIGURE 5, a folder 400 is a "my documents" folder. At a first level, the folder 400 includes folders 410, 420, and 430, corresponding to Clients 1, 2, and 3, respectively. At a second level, each of the folders 410, 420, and 430 contain a folder 411, 421, and 431, respectively, which each correspond to the contracts for the selected client. At a third level, each of the folders 411, 421, and 431 contains a folder 412, 422, and 432, respectively, each corresponding to the year 2001. At the third level, each of the folders 411, 421, and 431 also contains a folder 413, 423, and 433, respectively, each corresponding to the year 2002.

It will be appreciated that a number of obstacles are presented to a user who wishes to navigate a physical folder file structure such as that illustrated in FIGURE 5. For example, if the user wishes to work with all of the contracts that the user has produced, the user will first need to navigate to the folder 411 to work with the contracts for Client 1, and then will have to renavigate to the folder 421 to reach the contracts for Client 2, and will again have to renavigate to the folder 431 for the contracts for Client 3. This arrangement makes it difficult for the user to access all of the contracts, and in general prevents simultaneous viewing and manipulation of all of the contracts. Similarly, if the user wishes to view all of the contracts produced in the year 2001, the user will have to navigate and renavigate to the folders 412, 422, and 432, respectively. As will be described in more detail below, the virtual folders of the present invention provide an improved file system structure.

FIGURE 6 is a tree diagram of a virtual folder structure. As will be described in more detail below, virtual folders create location-independent views that allow users to manipulate their files and folders in convenient ways. As shown in FIGURE 6, the virtual folders are represented as stacks. A virtual folder 500 is an "all items" folder. At a first level, the virtual folder 500 contains virtual folders 510, 520, and 530, corresponding to clients, contracts, and year, respectively. As will be described in more detail below, this structure allows a user to access files according to a desired parameter.

FIGURE 7 is a tree diagram of the virtual folder structure of FIGURE 6, wherein at a second level, the virtual folder 510 further includes virtual folders 511 and 512, which correspond to contracts and year, respectively. In other words, the clients stack of virtual folder 510 is further filtered by contracts and year. The process for determining which files and items are contained in each of the virtual folders will be described in more detail below.

FIGURE 8 is a tree diagram of the virtual folder structure of FIGURE 7, wherein at a third level, the virtual folder 511 contains a virtual folder 513, which corresponds to a year. In other words, the contracts stack of virtual folder 511 is further filtered by year. While the virtual folder structure for the virtual folders 510, 511, and 513 have been structured according to clients, contracts, and year, it will be appreciated that the virtual folders allow for other structuring sequences to occur, as will be described in more detail below with reference to FIGURE 9.

FIGURE 9 is a tree diagram of the virtual folder structure of FIGURE 6, wherein at a second level, the virtual folder 520 has been further filtered into virtual folders 521 and 522, corresponding to clients and year. At a third level, the virtual folder 521 has further been filtered to a virtual folder 523, corresponding to a year. The contrast between the organizational structures of FIGURES 8 and 9 helps illustrate the flexibility of the virtual folder system. In other words, in a virtual folder system, a user is able to navigate the virtual folders according to desired parameters, as opposed to being dependent on the location-dependent views of a physical file structure such as that illustrated in FIGURE 5.

FIGURE 10 is a diagram illustrative of a screen display 600 showing the stacks of a document library. As noted above, stacks can be used to represent a type of virtual folder. As will be described in more detail below, the screen display 600 includes quick link elements 610-613, filter elements 620-626, activity elements 630-633, information and control elements 640-645, and virtual folder stacks 651-655.

The quick link elements include an "all categories" quick link 610, on "all authors" quick link 611, a "January work" quick link 612, and a selection for displaying additional quick links 613. As will be described in more detail below, quick links can be selected by a user to perform desired navigations of the virtual folders. Quick links may be provided by the system, and some quick links may be created and saved by a user.

The filter elements include a "filter by" indicator 620, an entry blank 621, a "by date" indicator 622, a "year" selector 623, a "pick an author" selector 624, a "pick a category" selector 625, and a "more filters" selector 626. The "filter by" indicator 620 directs a user to the fact that the items below can be used to filter the virtual folders or items. The entry blank 621 provides an area in which a user can type a desired new filter term. The "by date" indicator 622 directs a user to the fact that by selecting a date from the "year" selector 623, the virtual folders or items can be filtered by the selected year. The "pick an author" selector 624 allows a user to filter according to a specific author. The "pick a category" selector 625 allows a user to filter according to a selected category. The "more filters" selector 626 allows a user to pull up additional filters on the display.

The activity selectors include a "create a new category" selector 630, "activity" selectors 631 and 632, and a "more activities" selector 633. As will be described in more detail below, the activities that are presented may be for generally desirable functions, or may more specifically be directed to activities useful for the type of virtual folders that are currently being displayed. For example, the "create a new category" selector 630 can be selected by the user to create a new category which will be represented by a new stack.

As noted above, the activity selectors 631 and 632 may be more specifically directed to the type of folders or items that are being displayed. For example, the present display is of a document library, for which the "activity" selectors 631 and 632 may be directed to activities specifically tailored for documents, such as editing or creating attachments. If the present library had been a photo library, the "activity" selector 631 and 632 could be for activities specifically directed to photos, such as forming photo albums or sharing photos with other users.

The information and control elements include information lines 640 and 641, a control line 642, a backspace control 643, and information lines 644 and 645. The information lines 640 and 641 provide information as to the current navigation of the virtual folders or items. In the present example, the information line 640 indicates that the current navigation is to a document library, while the information line 641 indicates the more complete navigation, showing that the document library is within the storage area. The control line 642 provides a number of standard controls, and the backspace button 643 allows a user to back up through a navigation. The information line 644 provides numerical information about the contents of the present navigation. In the present example, the information line 644 indicates that there are 41 items which take up

100 MB in the stacks of the document library. The information line 645 is available to provide additional information, such as additional information about a file that is selected.

The stacks of the document library include an "ABC Corp." stack 651, a "backups stack" 652, a "business plans" stack 653, an "XYZ Corp." stack 654, and a "marketing reports" stack 655. The numbers on top of each of the stacks indicate how many items are in each stack. For example, the "ABC Corp." stack 651 is shown to include 8 items. The total number of items of the stacks adds up to the number of items indicated in the information line 644, which as described above is 41 in the present example. A selection box SB is provided which can be utilized by a user to select a desired item. The selection of the "ABC Corp." stack 651 yields a view of the items of that stack, as will be described below with respect to FIGURE 11.

FIGURE 11 is a diagram illustrative of a screen display showing the items in the "ABC Corp." stack 651 of FIGURE 10. It should be noted that the information lines 640 and 641 now indicate that the present navigation is showing the "ABC Corp." stack. The "ABC Corp." stack 651 is shown to include 8 documents 751-758, corresponding to documents 1-8, respectively. The information line 644 correspondingly indicates that there are 8 items which take up 20 MB of memory. Documents of FIGURE 11 may be further arranged into stacks within the ABC Corp. stack. In other words, within the virtual folder represented by the ABC Corp. stack 651, additional virtual folders may be organized to hold the documents, as will be described below with respect to FIGURES 12-16.

FIGURE 12 is a diagram illustrative of a screen display in which a stacking function is selected for the documents of FIGURE 11. As shown in FIGURE 12, the user is able to pull up a function box 760. The function box 760 includes a "view" selection 761, an "arrange icons by" selection 762, a "stacks" selection 763, a "refresh" selection 764, an "open containing folders" selection 765, a "cut" selection 766, a "copy" selection 767, an "undo" selection 768, a "new" selection 769, and a "properties" selection 770. The selection box SB is shown to be around the "stacks" selection 763.

FIGURE 13 is a diagram illustrative of a screen display in which a "stack by author" parameter is selected for the stacking function of FIGURE 12. As shown in FIGURE 13, a box 780 is displayed which presents various stacking options. The stacking options include an "unstack" option 781, a "stack by category" option 782, a

"stack by author" option 783, and a "stack by a user" option 784. The selection box SB is shown to be around the "stack by author" option 783.

FIGURE 14 is a diagram illustrative of a screen display in which the files of FIGURE 13 have been stacked by author. As shown in FIGURE 14, stacks 791 and 792 correspond to authors Bob and Lisa, respectively. As indicated by the numbers on top of each of the stacks, the Bob stack 791 includes two items, while the Lisa stack 792 includes five items. The item 758 (corresponding to document 8) did not have an author, and so is not included in an "author" stack. The stacks 791 and 792 illustrate that stacks may be organized at multiple levels, such as within the "ABC Corp." stack 651. Thus, the virtual folders may be formed at multiple levels, such as the "Lisa" stack 792 being within the "ABC Corp." stack 651 which is within the document library.

FIGURE 15 is a diagram illustrative of a screen display in which a "stack by category" option is further selected for restacking the files of FIGURE 14. As shown in FIGURE 15, the selection box SB is around the "stack by category" option 782. Since some of the items are already stacked in the stacks 791 and 792, the selection of the "stack by category" option 782 will restack the items, as will be described in more detail below with reference to FIGURE 16.

FIGURE 16 is a diagram illustrative of a screen display in which the files of FIGURE 14 are restacked by category. As shown in FIGURE 16, the stacks 793 and 794 correspond to the "XYZ Corp." and "marketing reports" categories, respectively. The items 751 and 752, corresponding to documents 1 and 2, were not designated for any additional categories, and thus did not fall into any of the other category stacks.

FIGURE 17 is a diagram illustrative of a screen display in which a quick link for physical folders is selected. The selection box SB is shown to be around the "all folders" quick link 616. As will be described in more detail below with respect to FIGURE 18, the "all folders" quick link 616 provides for switching to a view of physical folders.

FIGURE 18 is a diagram illustrative of a screen display showing physical folders. The physical folders that are shown contain the files of the virtual folder stacks of FIGURE 17. In other words, the items contained within the stacks 651-655 of FIGURE 17 are also contained in certain physical folders in the system. These are shown in FIGURE 18 as a "My Documents" folder 851 that is located on the present computer, a "Desktop" folder 852 that is located on the present computer, a "Foo" folder 853 that is located on the hard drive C:, a "My Files" folder 854 that is located on a server, an

"External Drive" folder 855 that is located on an external drive, a "My Documents" folder 856 that is located on another computer, and a "Desktop" folder 857 that is located on another computer.

As shown in FIGURE 18, a user is able to switch from the virtual files representation of FIGURE 17 to the physical file representation of FIGURE 18. This allows a user to toggle between virtual file representations and physical file representations, depending on which is desired for a current task. The different locations of the physical folders 851-857 also illustrate that the scope of the virtual file system may be relatively broad, as will be described in more detail below.

FIGURE 19 is a flow diagram illustrative of a routine 880 by which a user can directly manipulate virtual folders. As will be described in more detail below, the mechanisms that are provided for manipulating the virtual folders are similar to those that are currently used for manipulating regular folders (e.g., clicking and dragging, copying, pasting, etc.). As shown in FIGURE 19, at a block 882, the system provides defined actions that the user can perform for direct manipulation of the virtual folders that are represented as display objects. At a block 884, the user performs a defined action. As noted above, one example of this might be a user clicking and dragging a virtual folder to copy its contents to another virtual folder. At a block 886, the virtual folder and/or contents are manipulated as directed by the action performed by the user.

FIGURE 20 is a diagram illustrative of a screen display in which a new West Coast stack 656 has been added to the stacks of FIGURE 10. The West Coast stack 656 was formed by a user creating a new category of "West Coast." Upon its initial creation, the new West Coast stack 656 would be empty and have zero items. In the embodiment of FIGURE 20, two items have been added to the West Coast stack 656. One method for adding items to a stack is to select a particular item, and either modify or add additional categories to the category metadata for the item, such as adding the category "West Coast" to two items as was done in the embodiment of FIGURE 20. This process illustrates that the category data is a metadata property for an item that is a type of ad-hoc property. In other words, a property of this type does not have any implicit meaning, and can be assigned an arbitrary value by the user. For example, the category "property" can have any value whereas the "author" property should be the name of a person. As will be described in more detail below with reference to FIGURE 21, items may also be clicked and dragged to be copied from other stacks to the West Coast stack 656 (in which case

the categories of the items are automatically updated to include "West Coast"). In this regard, FIGURE 20 shows that the selection box SB is around the ABC Corp. stack 651, in preparation for its contents being copied.

FIGURE 21 is a diagram illustrative of a screen display in which direct manipulation is used for copying the files from the ABC Corp. stack 651 to the West Coast stack 656. In other words, as shown in FIGURE 20, the user selected the ABC Corp. stack 651, and then as shown in FIGURE 21 the user has clicked and dragged the stack to be copied to the West Coast stack 656. Thus, the West Coast stack 656 which had two items in FIGURE 20, is now shown to include a total of ten items, including the additional eight items from the ABC Corp. stack 651. When the items from the ABC Corp. stack 651 were copied to the West Coast stack 656, this was accomplished by modifying the category descriptions of the eight items to also include the "West Coast" category in addition to including the original "ABC Corp." category. This illustrates one type of direct manipulation that may be performed.

Another example of direct manipulation is right clicking an item and selecting delete. In one embodiment, when a deleting function is selected by a user, the user is queried whether the item should be deleted all together, or simply removed from the present virtual folder. If the item is just to be removed from a present virtual folder category stack as noted above, this can be accomplished by removing the desired category from the metadata for the item. In other words, if one of the items that had been copied from the ABC Corp. stack 651 to the West Coast stack 656 was then to be removed from the West Coast stack 656, this could be accomplished by modifying the category data for the particular file to no longer include the "West Coast" category.

FIGURE 22 is a flow diagram illustrative of a routine 900 for the system dynamically generating new filter terms. Filter terms are utilized for manipulating the virtual folders. The filtering terms are essentially utilized as a set of tools for narrowing down a set of items. In one embodiment, filters consist of metadata categories and their values (presented to the user in the user interface as clickable links or drop-down menus). The user clicks on a filter term in order to filter down the current results set of items on the display.

FIGURE 22 illustrates how filters may be dynamically generated. As shown in FIGURE 22, at a block 902, the properties (from the metadata) of the items in a collection on the present display are reviewed. In a block 904, proposed filter terms are

dynamically generated based on common properties of the items. At a block 906, the proposed filter terms are presented to the user for possible selection for filtering items. As an example of this process, the system may review the properties of a set of items, and if the items generally have "Authors" as a property, the filter can provide a list of the authors to filter by. Then, by clicking on a particular Author, the items that don't have that Author are removed from the set on the display. This filtering process provides the user with a mechanism for narrowing the set of items on the display.

FIGURE 23 is a flow diagram illustrative of a routine 920 for the system filtering items based on the selection of a filter term. At a block 922, the user either enters a new filter term or else selects one of the filter terms that have been presented by the system. As noted above, the filter terms may be dynamically generated by the system, or they may be preset. At a block 924, the items from the collection on the display are evaluated with regard to whether their selected properties match the filter term. For example, if the filter term is for items that were authored by "Bob," then the items are evaluated in accordance with whether their author property includes "Bob". At block 926, the items for which the selected properties do not match the filter term are removed from the collection on the display.

FIGURE 24 is a diagram illustrative of a screen display in which the stacks of FIGURE 10 have been filtered by the term "AB". As shown, in the filter area 621, the term "AB" has been typed by a user. The information lines 640 and 641 indicate that the items in the display are now those that have been filtered by the term "AB". As shown, the ABC Corp. stack 651 still contains eight items, while the Backups stack 652 now contains three items, and the XYZ Corp. stack 654 also contains three items. The information line 644 thus indicates that there are a total of 14 items, taking up a total of 35 MB of memory.

FIGURE 25 is a diagram illustrative of a screen display in which the stacks of FIGURE 10 have been filtered by the term "ABC". With regard to the filter term "AB" of FIGURE 24, the user has simply typed the additional letter "C" to make the total filter term "ABC". As shown in FIGURE 25, the information lines 640 and 641 now indicate that the items on the display are those that contain the term "ABC". The ABC Corp. stack 651 is still shown to contain eight items, while the Backups stack 652 now contains only two items. The information line 644 now indicates that there are a total of 10 items in the stacks on the display, which take up a total of 25 MB of memory. FIGURES 24

and 25 thus provide examples of how a user may enter new filter terms, and how those filter terms are then used to filter the items that are shown on the display.

FIGURE 26 is a diagram illustrative of a screen display in which the system provided filter term "year 2002" is selected. As noted above, under the by date indicator 622, the year selections 623 include the years 2000, 2001, or 2002. The selection box SB is shown to be around the year 2002, indicating that the user is selecting that as the desired filter term.

FIGURE 27 is a diagram illustrative of a screen display in which the filter term "2002" has been applied. Also shown is the further selection of the "pick a month" selector 623A. As shown in FIGURE 27, after applying the filter term "2002", the number of items in the stacks have been reduced. More specifically, the ABC Corp. stack 651 now contains six items, the Backups stack 652 now contains eight items, the Business Plans stack 653 now contains three items, and the XYZ Corp. stack 654 now contains five items. The information line 644 now indicates a total of 22 items, taking up a total of 50 MB of memory. The information lines 640 and 641 now indicate that the items shown on the display are those that have been filtered to contain the filter term "2002".

FIGURE 28 is a diagram illustrative of a screen display in which a list is presented for selecting a month for filtering. A box 950 is provided which includes the list of the months. The box 950 has been provided on the display due to the user selecting the "pick a month" selector 623A. The selection box SB is shown to be around the month of January.

FIGURE 29 is a diagram illustrative of a screen display wherein the stacks of FIGURE 28 have been further filtered by the month of January, and further showing a filter term of "day". As shown in FIGURE 29, the information lines 640 and 641 now indicate that the items on the display are those that have been filtered by the term "January". The Backups stack 652 is now shown to contain two items, while the Business Plans stack 653 is also shown to contain two items. The information line 644 indicates that there are a total of four items on the display, which take up a total of 10 MB of memory. A "pick by day" selector 623B is provided, should the user wish to further filter the results to a specific day.

FIGURE 30 is a flow diagram illustrative of a routine 940 for creating a new quick link. As will be described in more detail below, quick links are predefined links

that can be clicked on by a user to create user selected views of the sets of items. In one embodiment, a quick link may be thought of as a type of pivot. Quick links provide a mechanism for retrieving a virtual folder. Clicking a quick link can take a user to a desired folder (in the same way that clicking a "favorites" may take a user to a Web site. The quick links can be predefined by the system, or can be set by a user. For example, clicking on "all authors" could return a view stacked by authors. Clicking on "all documents" may return a flat view for all of the documents for all of the storage areas. Users can also create their own quick links.

As shown in FIGURE 30, at a block 942, a user makes a selection on the display to indicate that a new quick link should be formed from the present filter term or navigation. At a block 944, the user provides a new name for the new quick link. At a block 946, the new quick link is saved and the new quick link name is provided in the quick link section on the display.

FIGURE 31 is a diagram illustrative of a screen display for creating a new quick link called "January Work" based on the filtering of FIGURE 29. As described above, in FIGURE 29, the stacks have been filtered by the month of January. In FIGURE 31, the user has indicated that the filtering of FIGURE 29 should be saved as a new quick link, and has named the new quick link "January work". Thus, the new January work quick link 612 is shown in the quick links section of the display. With regard to forming new quick links, the user is generally provided with an option such as "save this collection as a quick link".

FIGURE 32 is a diagram illustrative of a screen display in which a quick link of "All Authors" is selected. As shown in FIGURE 32, the selection box SB is shown around the All Authors selection 611. Other examples of collections that might be accessible by quick links include "all authors", "recent documents", "all documents I've shared", "all documents I've authored", "all documents not authored by me", "desktop", and "all types".

FIGURE 33 is a diagram illustrative of a screen display in which a list of all of the authors of the items of FIGURE 32 is presented. As shown in FIGURE 33, an information line 950 is provided, which indicates columns for showing the name of an item, the author, the modified date, the type, the size, and the location of an item. A list of Authors 951-954 are shown, corresponding to Authors 1-4, respectively.

FIGURE 34 is a diagram illustrative of a screen display in which "Author 1" has been selected from the list of FIGURE 33. The Author 1's documents include documents 951A and 951B, corresponding to documents 1 and 2, respectively. The document 951A is shown to have been authored by Author 1, was modified on 11 July, 2001, is a Microsoft Excel file, takes up 282 Kb of memory, and was obtained from the location \\server1\folder2. The document 951B is shown to have been authored by Author 1, was modified on 22 December, 2002, is a Microsoft Word file, takes up 206 kilobytes of memory, and is physically stored in the location My Documents\folder1. The locations of the documents 951A and 951B also illustrate that the virtual folders of the present invention may contain items from different physical locations, as will be described in more detail below.

FIGURE 35 is a flow diagram illustrative of a routine 960 for creating a new library. One example of a library is the documents library described above with reference to FIGURE 10. In general, libraries consist of large groups of usable types of files that can be associated together. For example, photos may be one library, music may be another, and documents may be another. Libraries may provide tools and activities that are related to the particular types of items. For example, in the photo library, there may be tools and filters that relate to manipulating photos, such as for creating slide shows or sharing pictures. As shown in FIGURE 35, at a block 962, a new library is created which is to include items with selected characteristics. At a block 964, the selected items are grouped into the library. At a block 966, the tools and/or activities related to the selected characteristics of the items or to other desired functions are provided.

FIGURE 36 is a diagram illustrative of a screen display in which a collection of available libraries are shown. As shown in FIGURE 36, the libraries include a documents library 971, a photos and video library 972, a music library 973, a messages library 974, a contacts library 975, and a TV and movies library 976, as well as an all items library 977. The all items library 977 is shown to include 275 items, which is the total number of items from all of the other libraries combined. The information line 644 indicates a total of 275 items, which take up a total of 700 MB of memory. It should be noted that the documents library 971 is the library that was described above with respect to FIGURE 10.

FIGURE 37 is a flow diagram illustrative of a routine 990 for defining the scope of a virtual folder collection. As will be described in more detail below, a virtual folder

system is able to represent items from multiple physical locations (e.g., different hard drives, different computers, different networks locations, etc.) so that to a user, all of the items are readily accessible. For example, a user can be presented with music files from multiple physical locations on a single display, and manipulate the files all at once.

As shown in FIGURE 37, at a block 992, a scope is defined for the physical locations from which items are to be drawn. At a block 994, in response to a query, the items are drawn from the physical locations as defined in the scope. At a block 996, all of the items drawn by the query are presented on a single display.

FIGURE 38 is a block diagram illustrative of the various sources which may form the scope of a virtual folder collection. As shown in FIGURE 38, the system 1000 may include a present computer 1010, an additional computer 1020, external and removable storage 1030, and locations on a network 1040. The overall scope 1001 is described as including all of the physical locations from which a user's items are drawn to create collections. The scope may be set and modified by a user. As noted above, other figures have illustrated that items may come from different physical locations, such as FIGURE 34 showing different documents coming from a server and a My Documents folder on a present computer, and in FIGURE 18 showing physical folders that are physically stored in multiple locations.

FIGURE 39 is a flow diagram illustrative of a routine 1080 for including non-file items in a virtual folder collection. Non-file items are contrasted with file items that are typically located in a physical file storage. Examples of non-file items would be things like e-mails, or contacts. As shown in FIGURE 39, at a block 1082 a database is utilized to include non-file items along with file items that may be searched by a query. At a block 1084, in response to a query, both non-file items and file items are drawn to match the query. At a block 1086, both the non-file items and the file items that matched the query are presented on the display.

FIGURE 40 is a diagram illustrative of a screen display showing various non-file items. As shown in FIGURE 40, the items have been filtered to those that include "John". The items are shown to include a contact item 1101, an e-mail item 1102, and document items 1103 and 1104. The contact item 1101 and e-mail item 1102 are non-file items. The present system allows such non-file items to be included with regular file items, such that they can be organized and manipulated as desired by a user. As was described above with respect to FIGURE 2, such non-file items may be contained entirely

within the relational database 230, which otherwise includes information about the properties of files.

While the preferred embodiment of the invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. In a computer system having a display and a memory for storing items, the items having associated metadata properties, a method of exposing the items to a user, the method comprising:

selecting a first metadata property;

searching for items that have the selected first metadata property; and

providing a first virtual folder display object that represents the collection of items that have the first metadata property.

2. The method of Claim 1, wherein the virtual folder display object is represented as a stack.

3. The method of Claim 1, wherein a second metadata property is selected to create a second virtual folder within the first virtual folder, the items that are in the first virtual folder being further searched to determine if they have the second metadata property, the second virtual folder representing the items within the first virtual folder that have the second metadata property.

4. The method of Claim 1, wherein a plurality of virtual folder display objects are provided on the display, each virtual folder display object representing a collection of items that have a selected metadata property.

5. The method of Claim 4, wherein a plurality of the items are also physically stored in physical folders.

6. The method of Claim 5, wherein a user can select to view the physical folders.

7. The method of Claim 1, wherein a selected metadata property can be added to selected items.

8. The method of Claim 7, wherein the addition of a metadata property to an item also causes the item to be added to a virtual folder that is based on the selected metadata property.

9. The method of Claim 8, wherein the selected metadata property is a specified category.

10. The method of Claim 1, wherein the user can make changes to the virtual folder through direct manipulation of the virtual folder display object.

11. The method of Claim 10, wherein at least one type of direct manipulation comprises clicking and dragging a virtual folder.

12. The method of Claim 11, wherein clicking and dragging can be used to copy the items from a first virtual folder to a second virtual folder, and the copying is implemented at least in part by adding the selected metadata property of the second virtual folder to each of the items from the first virtual folder.

13. The method of Claim 1, wherein a filter is provided on the display that can be selected by a user to filter items based on the metadata property specified by the filter.

14. The method of Claim 1, wherein a quick link is provided on the display that can be selected by a user to view a specified collection of items.

15. The method of Claim 1, wherein a library is provided that includes a collection of items and a set of tools for manipulating the items of the library.

16. The method of Claim 1, wherein the items in a virtual folder may be physically stored in memories at different physical locations.

17. The method of Claim 16, wherein the different physical locations comprise a present computer and at least one of a different computer, a location on a network, and an external storage device.

18. The method of Claim 1, wherein the items comprise both file items and non-file items.

19. The method of Claim 18, wherein the non-file items comprise at least one of contacts or e-mails.

20. A computer-readable medium having computer-executable components for implementing a method of exposing items to a user, the items having associated metadata properties, the method comprising:

selecting a first metadata property;

searching for items that have the selected first metadata property; and

providing a first virtual folder display object that represents the collection of items that have the first metadata property.

21. The method of Claim 20, wherein the virtual folder display object is represented as a stack.

22. The method of Claim 20, wherein a second metadata property is selected to create a second virtual folder within the first virtual folder, the items that are in the first virtual folder being further searched to determine if they have the second metadata property, the second virtual folder representing the items within the first virtual folder that have the second metadata property.

23. The method of Claim 20, wherein a selected metadata property can be added to selected items.

24. The method of Claim 20, wherein the user can make changes to the virtual folder through direct manipulation of the virtual folder display object.

25. The method of Claim 20, wherein the items in a virtual folder may be physically stored in memories at different physical locations.

26. The method of Claim 20, wherein the items comprise both file items and non-file items.

27. In a system for displaying items, the system comprising:

means for selecting a first metadata property;

means for searching for items that have the selected first metadata property; and

means for providing a first virtual folder display object that represents the collection of items that have the first metadata property.

28. The system of Claim 27, further comprising means for creating a second virtual folder within the first virtual folder.

29. The system of Claim 27, further comprising means to add a selected metadata property to a selected item.

30. The system of Claim 29, wherein the means to add a metadata property to an item comprises means for selecting a metadata property of an item and entering a new property.

31. The system of Claim 27, further comprising means for implementing direct manipulation of virtual folder display objects.

32. The system of Claim 27, further comprising means for including items from different physical locations in the search for items that have the selected first metadata property.

33. The system of Claim 27, further comprising means for including both file items and non-file items in the search for items that have the selected first metadata property.

34. A system for exposing items that are stored in a memory to a user, comprising:

a folder processor that obtains queries from a user;

a relational database for storing information about items; and

wherein:

the folder processor obtains a query from a user and passes the query to the relational database;

the relational database provides results back to the folder processor; and

based on the results from the relational database, the folder processor provides the results to the user as virtual folders.

35. The method of Claim 34, wherein the query is a default query that is taken from a registry.

36. The method of Claim 34, wherein the query is obtained from a user that has decided to change the pivot of the virtual folder view.

37. The method of Claim 34, wherein the query that is passed to the relational database comprises a database query object that is constructed by the folder processor.

38. The method of Claim 34, wherein the results that are provided back to the folder processor comprise database rows and columns.

39. The method of Claim 34, wherein after receiving the results from the relational database the folder processor takes the results and converts them into an enumerator structure.

40. The method of Claim 39, wherein the enumerator structure is used to populate the display with the resulting virtual folders.

41. A system for displaying items, comprising:

a folder processor means;

a relational database means;

wherein:

the folder processor means obtains a query from a user and passes the query to the relational database;

the relational database means provides results back to the folder processor; and

based on the results from the relational database means, the folder processor means provides the results to the user as virtual folders.

42. The system of Claim 41, further comprising means for obtaining a query from a user.

43. The system of Claim 41, further comprising means for constructing a database query object that is included in the query that is passed to the relational database.

44. The system of Claim 41, further comprising means for providing database rows and columns as the results that are provided back to the folder processor.

45. The system of Claim 41, further comprising means for converting the results from the relational database into an enumerator structure.

46. The system of Claim 45, further comprising means to populate a display with virtual folders which result from the enumerator structure.

47. In a computer system having a display and a memory for storing items, a method for manipulating the items, comprising:

representing groups of items as virtual folders; and

providing at least a first defined action that can be performed for direct manipulation of a virtual folder, wherein when the first defined action is performed, the virtual folder is manipulated as directed by the performed action.

48. The method of Claim 47, wherein the defined action is clicking and dragging the virtual folder.

49. The method of Claim 48, wherein clicking and dragging a first virtual folder to a second virtual folder performs the function of copying the items of the first virtual folder to the second virtual folder.

50. The method of Claim 49, wherein items are contained within a virtual folder on the basis of having a selected metadata property that is the basis for the virtual folder, and the action of copying the items from a first virtual folder to a second virtual folder is implemented at least in part by adding the metadata property of the second virtual folder to each of the items from the first virtual folder.

51. The method of Claim 50, wherein the metadata property that is added to the items of the first virtual folder is a category.

52. A computer-readable medium having computer-executable components for implementing a method for manipulating items, comprising:

representing groups of items as virtual folders; and

providing at least a first defined action that can be performed for direct manipulation of a virtual folder, wherein when the first defined action is performed, the virtual folder is manipulated as directed by the performed action.

53. The method of Claim 52, wherein the defined action is clicking and dragging the virtual folder.

54. The method of Claim 53, wherein clicking and dragging a first virtual folder to a second virtual folder performs the function of copying the items of the first virtual folder to the second virtual folder.

55. The method of Claim 54, wherein items are contained within a virtual folder on the basis of having a selected metadata property that is the basis for the virtual folder, and the action of copying the items from a first virtual folder to a second virtual folder is implemented at least in part by adding the metadata property of the second virtual folder to each of the items from the first virtual folder.

56. The method of Claim 55, wherein the metadata property that is added to the items of the first virtual folder is a category.

57. A system for manipulating items, comprising:

means for representing groups of items as virtual folders; and

means for implementing defined direct manipulation actions for the manipulation of the virtual folders.

58. The system of Claim 57, wherein the means for implementing direct manipulation includes means for implementing clicking and dragging of a virtual folder.

59. The system of Claim 58, wherein clicking and dragging a first virtual folder to a second virtual folder performs the function of copying the items of the first virtual folder to the second virtual folder.

60. The system of Claim 59, wherein items are contained within a virtual folder on the basis of having a selected metadata property that is the basis for the virtual folder, and the action of copying the items from a first virtual folder to a second virtual folder is implemented at least in part by adding the metadata property of the second virtual folder to each of the items from the first virtual folder.