SYSTEM AND METHOD FOR MANAGING DATA USING STATIC LISTS

FIELD OF THE INVENTION

In general, the present invention relates to data storage systems and, in particular, to systems and methods for managing data using static lists.

BACKGROUND OF THE INVENTION

As the use of electronic media to store text, music, pictures, and other types of data grows and the restrictions on data storage capacities lessen, computer users find themselves faced with enormous numbers of files to manage. Conventional file systems, such as those based on a file allocation table, or FAT file system, can make management of files difficult. For example, the traditional directory access to files that is provided with conventional file systems assumes that the users wishes to maintain their files in a hierarchical directory tree. However, besides being location dependent, a hierarchical organization may not be the most advantageous way to access the files from the user's point of view.

In the context of the Windows® operating system user interface, one technique for making access to files easier is the shortcut. A shortcut that provides a link to a file may be created on the desktop or in a folder, and is a quick way to start a program or open a file or folder without having to go to its permanent location. But shortcuts may not be reliable since they are not updated to reflect changes in the location or status of the underlying file. For example, moving the file to a different directory results in an error when accessing the shortcut.

Another technique for making access to files easier is the playlist. Media players offer users playlists as a way to organize certain types of files for later playback. For example, in the Windows Media Player®, the playlist contains references to music files for playback through the media player in a designated order. But playlists suffer from the same drawback as shortcuts in that the references in the playlist are not updated to reflect changes in the location or status of the underlying files. If a music file is moved or deleted, the user must hunt through all of his or her playlists to update or remove the outdated references.

Both the shortcut and playlist model of accessing files are further limited by their inability to provide to the user with alternative ways to access items other than through another folder, or in a certain order.

SUMMARY OF THE INVENTION

To overcome the above-described problems, a system, method, and computer-accessible medium for managing data using static lists are provided. Static lists facilitate arbitrary grouping of items of data independent of their locations and in ways that are meaningful to the user.

In accordance with one aspect of the present invention, a static list is a set of items defined by a root item, a direction, and the entry relationships with that root item in that direction. The items in the set are determined by following the entry relationships with the root item. The direction is either to or from the root item, depending on whether the root item is the target or the source of the entry relationship. The static list also defines the properties that each entry relationship in the list is required to have.

In accordance with another aspect of the present invention, verbs are provided to manage a static list. A verb is an action that may be performed on the items in the static list, and includes, among others, move, copy, add, remove, and delete. The actions performed on the items include actions performed on the entry relationships between the item and the root item.

In accordance with a further aspect of the present invention, a view is provided to specify characteristics for displaying data from a static list, including visibility, order, and formatting, among other characteristics.

In accordance with yet another aspect of the present invention, using static lists, the user is able to propagate certain security attributes to the items in the list so that others may access them via the list. The user may also add other information to the list as metadata to enhance the usefulness of the list and the items contained therein.

In accordance with a still further aspect of the present invention, using static lists, each item in the list is automatically managed so that the references to the data are always valid, even when the location, status, or other characteristic of the data changes.

In accordance with yet other aspects of the present invention, a computer accessible medium for managing data using static lists is provided. The computer accessible medium comprises data and computer executable components to create and manage static lists. The data defines the static list and the items contained therein. The computer executable components are capable of performing actions generally consistent with the above-described method.

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 depiction of a conceptual storage model for managing data using static lists, formed in accordance with the present invention;

FIGURE 2 is a depiction of further aspects of the conceptual storage model for defining a static set, formed in accordance with the present invention;

FIGURE 3 is a depiction of further aspects of the conceptual storage model for defining a static list, formed in accordance with the present invention;

FIGURE 4 is a depiction of further aspects of the conceptual storage model for defining a list entry in a static list, formed in accordance with the present invention;

FIGURE 5 is a depiction of further aspects of the conceptual storage model for defining an exemplary static list, formed in accordance with the present invention;

FIGURE 6 is a depiction of a grid containing values for property names and list items depicted in the exemplary static list in FIGURE 5;

FIGURE 7 is a depiction of a user interface containing a set of items that may be used in a static list, formed in accordance with the present invention;

FIGURE 8 is a depiction of a user interface displaying an exemplary static list containing items depicted in FIGURE 7, as formed in accordance with the present invention;

FIGURE 9 is a depiction of a user interface displaying yet another exemplary static list that contains the exemplary static list depicted in FIGURE 8, as formed in accordance with the present invention;

FIGURE 10 is a depiction of further aspects of the conceptual storage model for defining a view that may be applied to a static list, formed in accordance with the present invention;

FIGURE 11 is a block diagram of a general-purpose computer system suitable for containing static lists, formed in accordance with the present invention;

FIGURE 12 is a flow diagram illustrating the logic performed by a general-purpose computer system for managing data using static lists, formed in accordance with the present invention;

FIGURE 13 is a flow diagram illustrating the logic performed by a general-purpose computer system for moving items between static lists, formed in accordance with the present invention;

FIGURE 14 is a flow diagram illustrating the logic performed by a general-purpose computer system for copying items between static lists, formed in accordance with the present invention;

FIGURE 15 is a flow diagram illustrating the logic performed by a general-purpose computer system for adding items to static lists, formed in accordance with the present invention;

FIGURE 16 is a flow diagram illustrating the logic performed by a general-purpose computer system for removing items from static lists, formed in accordance with the present invention;

FIGURE 17 is a flow diagram illustrating the logic performed by a general-purpose computer system for deleting items from static lists, formed in accordance with the present invention;

FIGURE 18 is a block diagram overview of an implementation of static lists formed in accordance with the present invention using XML files; and

FIGURE 19 is a block diagram overview of an implementation of static lists, formed in accordance with the present invention, using file system containers.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The following discussion is intended to provide a brief, general description of a computing system suitable for implementing various features of the invention. While the computing system will be described in the general context of a personal computer usable in a distributed computing environment, where complementary tasks are performed by remote computing devices linked together through a communication network, those skilled in the art will appreciate that the invention may be practiced with many other computer system configurations, including multiprocessor systems, minicomputers, mainframe computers, and the like. In addition to the more conventional computer systems described above, those skilled in the art will recognize that the invention may be practiced on other computing devices, including laptop computers, tablet computers, personal digital assistants (PDAs), and other devices upon which computer software or other digital content is installed.

While aspects of the invention may be described in terms of programs executed by applications in conjunction with a personal computer, those skilled in the art will recognize that those aspects also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc., which perform particular tasks or implement particular abstract data types.

FIGURE 1 is a depiction of a conceptual storage model for managing data using static lists formed in accordance with the present invention. An item 150 is a primary container of data. Each item contains multiple properties 130 to hold the data, and a reference to a type 120 that defines what properties the item has. An item 150 may contain additional properties 130 that are not defined by the item's type 120.

A relationship 110 is an association between two items. Each relationship refers to two items 150 called a source 102 or a target 104, depending on direction of the relationship 110. Source items 102 originate the relationship 110, and target items 104 receive the relationship 110.

An extension 140 is similar to an item 150, in that it contains properties 130 defined by a type 120. But extensions 140 are associated with exactly one item 150 and have different types 120.

The type 120 defines the structure of an item 150, relationship 110, or extension 140 by defining its properties. Since types 120 can be used with items, relationships, or extensions, they are commonly referred to as item types, relationship types, or extension types.

FIGURE 2 is a depiction of further aspects of the conceptual storage model for defining a static set formed in accordance with the present invention. Specifically, FIGURE 2 depicts a static set 220. As illustrated, a static set 220 is a set 210 that explicitly associates an item 150 with other items using relationship types 230. A static set 220 contains a root item 150, a relationship type 230, and a particular direction of the root item to the associated items, either a target direction 104 or a source 102 direction. The static set's membership is determined by following relationships of the given type that have the root item as either the source or the target, depending on the direction specified in the set.

Any combination of an item 150, a relationship type 230, and a direction 102/104 determines a static set 220. For example, the set of authors of a document can be found

by following author relationships from the document root item, as can the set of document authored by a person by following the same relationship in the other direction.

FIGURE 3 is a depiction of further aspects of the conceptual storage model for defining a static list formed in accordance with the present invention. A static list 310 is a type of static set 220 that allows users to organize items 150 into collections in arbitrary ways. The static list 310 comprises a list item 320 and a list entry relationship type 330. The list item 320 is the root item and the list entry relationship type 330 defines the set of properties 130 and direction 102/104 that each relationship 110 in the list must have in an entry template 340 that specifies property names 350 for each of the properties 130. Because each relationship 110 has the same properties 130, the static list 310 can be thought of as a table or grid where each entry relationship 110 is a row, and each property 130 is a column, an example of which is described below with reference to FIGURES 5 and 6.

FIGURE 4 is a depiction of further aspects of the conceptual storage model for defining a list entry in a static list formed in accordance with the present invention. Specifically, a list entry 410 is a relationship 110 that has one or more properties 130. FIGURE 5 is a depiction of further aspects of the conceptual storage model for defining an exemplary static list formed in accordance with the present invention. The list item 320 that is the root item of the static list 310 is the source 102 of two list entry relationships 510A and 510B that associate the list item 320 with target items picture A 520A, and picture B 520B. Each relationship 510A and 510B has two properties 530 and 540, as defined in entry template 340. Properties 530A and 530B are comments to the referenced pictures--picture A 520A and picture B 520B--having respective values "What a long neck!" and "Say Cheese!" Properties 540A and 540B are ratings of the referenced pictures--picture A 520A and picture B 520B--having respective values "4" and "5."

FIGURE 6 is a depiction of a grid containing values for property names and list items depicted in the exemplary static list in FIGURE 5. As shown, the list items 320 comprising picture A 650 and picture B 660 form the rows of the grid 600, and the property names 350 associated with those list items 320 are the columns of the grid 600. The property names 350 are based on the entry template 340, and in the illustrated example, are comment 620 and rating 630, as previously described. Other property names 350 may be added as well, such as order 610, specifying the order in which the list

items 320 should be presented in a display, and any other info 640 that the user might deem useful for items of this type (e.g., where or when the picture was taken).

FIGURE 7 is a depiction of a user interface containing a set of items that may be used in a static list formed in accordance with the present invention. As shown, the set of items is for My Pictures 710, and contains six pictures--picture A 720A, picture B 720B, picture C 720C, picture D 720D, picture E 720E, and picture F 720F.

FIGURE 8 is a depiction of a user interface 800 displaying an exemplary static list 810 containing some of the items depicted in FIGURE 7 as formed in accordance with the present invention. Using the exemplary static list described in FIGURE 5, a static list labeled "good giraffe pictures" 810 is shown with two of the six pictures shown in FIGURE 6, including picture A 720A and picture B 720B, corresponding to target items 520A and 520B (FIGURE 5). The accompanying texts "what a long neck!" 820A and "say cheese!" 820B correspond to the comment properties 530A and 530B.

FIGURE 9 is a depiction of a user interface 900 displaying yet another exemplary static list 910 that contains the exemplary static list 810 depicted in FIGURE 8 as formed in accordance with the present invention. In addition, the static list 910 labeled "My Safari Notes" further contains texts "We saw giraffes . . ." 920X and "Then we saw elephants . . ." 920Y, which would correspond to properties 130 defined for the relationships 110 in static list 910, say, for example, a property 130 of note as defined by the entry template 340 for a root list item 320 for a safari journal. Here, the target items 150 are the pictures of the elephants--picture D 720D and picture F 720F--as well as the original static list 810 depicted in FIGURE 8. This illustrates that static lists can have target items that are actually other static lists, i.e., that static lists can be nested.

FIGURE 10 is a depiction of further aspects of the conceptual storage model for defining a view that may be applied to a static list formed in accordance with the present invention. A view 1010 is a collection of property infos 1020. The property infos 1020 specify a property name 1030 of a property 130, and the display characteristics for the corresponding properties 130 that are defined for the items 150 and relationships 110 that comprise the static list 310. The view 1010 is applied to a static list 310 by retrieving the properties 130 by property names 1030 and applying the display characteristics to the values of the properties 130 in preparation for incorporating the values into a user interface to display the list to the user, such as the user interfaces illustrated in FIGURES 8 and 9.

FIGURE 11 is a block diagram of a general-purpose computer system suitable for containing static lists formed in accordance with the present invention. The system 1100 includes a personal computer 1102 comprising a processing unit 1122, a system memory 1124, and a system bus 1126 that couples the system memory to the processing unit 1122. The system memory 1124 includes read-only memory (ROM) 1128 and random-access memory (RAM) 1130. A basic input/output system 1132 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 1102, such as during startup, is stored in ROM 1128. The personal computer 1102 further includes a hard disk drive 1134, a magnetic disk drive 1138, e.g., to read from or write to a removable disk 1140, and an optical disk drive 1142, e.g., for reading a CD-ROM disk 1144 or to read from or write to other optical media. The hard disk drive 1134, magnetic disk drive 1138, and optical disk drive 1142 are connected to the system bus 1126 by a hard disk drive interface 1154, a magnetic disk drive interface 1156, and an optical drive interface 1160, respectively. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer 1102. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk, and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media that are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, ZIP disks, and the like may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 1130, including an operating system 1146, one or more application programs 1148, other program modules 1150, such as the extensions and interfaces of the present invention, and program data 1152, including the command item and insert location data of the present invention. A user may enter commands and information into the personal computer 1102 through input devices such as a keyboard 1160 or a mouse 1162. Other input devices (not shown) may include a microphone, touch pad, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 1122 through a user input interface 1164 that is coupled to the system bus, but may be connected by other interfaces (not shown), such as a game port or a universal serial bus (USB). A display device 1190 is also connected to the system bus 1126 via a display subsystem that typically includes a graphics display interface (not shown) and a code module, sometimes referred to as a display driver, to interface with the

graphics display interface. While illustrated as a stand-alone device, the display device 1190 could be integrated into the housing of the personal computer 1102. Furthermore, in other computing systems suitable for implementing the invention, such as a PDA, the display could be overlaid with a touch-screen. In addition to the elements illustrated in FIGURE 11, client devices also typically include other peripheral output devices (not shown), such as speakers or printers.

The personal computer 1102 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 1165. The remote computer 1165 may be a server, a router, a peer device, or other common network node, and typically includes many or all of the elements described relative to the personal computer 1102. The logical connections depicted in FIGURE 11 include a local area network (LAN) 1166 and a wide area network (WAN) 1167. The LAN 1166 and WAN 1167 may be wired, wireless, or a combination thereof. 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 1102 is connected to the LAN 1166 through a network interface 1168. When used in a WAN networking environment, the personal computer 1102 typically includes a modem 1169 or other means for establishing communications over the WAN 1167, such as the Internet. The modem 1169, which may be internal or external, is connected to the system bus 1126 via the user input interface 1164. In a networked environment, program modules depicted relative to the personal computer 1102, 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 communication link between the computers may be used. In addition, the LAN 1166 and WAN 1167 may be used as a source of nonvolatile storage for the system.

FIGURE 12 is a flow diagram illustrating the logic performed by a general-purpose computer system for managing data using static lists formed in accordance with the present invention. At processing block 1210, the user further defines which properties he or she wishes to require for each list entry relationship, i.e., the list entry template. Again, using the same example, the list entry relationship is defined to have at least two properties that describe the comment associated with the target pictures as well as a rating. At processing block 1220, a processor uses the definitions to generate a static

list, including generating the list entry relationships in response to user input to the entry template. For example, depending on the type of graphical user interface employed in the embodiment, the user might drag and drop the arbitrarily selected target pictures (here, picture A 520A/720A and picture B 520B/720B) from the user interface 710 depicted in FIGURE 7 into the user interface 810 depicted in FIGURE 8. This user action will trigger generation of the list entry relationships in the static list in accordance with the list entry template 340 (FIGURE 3 and FIGURE 5).

In one embodiment, processing continues at process block 1230, where the user can elect to apply a previously defined View 1010 to the list in order to display, at process block 1240, the list contents in a user interface such as the user interface 810 depicted in FIGURE 8.

FIGURE 13 is a flow diagram illustrating the logic performed by a general-purpose computer system for moving items between static lists formed in accordance with the present invention. The process begins at predefined process block 1310 invoked in response to a user input to move an item from one static list to another. Processing begins at process block 1320, where a processor deletes the list entry relationship associated with the selected item from the current static list. At processing block 1330, the processor creates a new instance of an entry relationship type for the new list. At decision block 1340, the process determines whether the type of entry relationship being created is the same as the type of entry relationship just deleted from the old list. If yes, the processing continues at process block 1350, where the processor copies the entry relationship's properties from old to new. So, for example, if the type of entry relationship is denoted as good giraffe pictures, then the same properties of comments and ratings will be copied to the new list.

FIGURE 14 is a flow diagram illustrating the logic performed by a general-purpose computer system for copying items between static lists formed in accordance with the present invention. The process begins at predefined process block 1410 invoked in response to a user input to copy an item from one static list to another. Similar to the move logic described in reference to FIGURE 13, at processing block 1420, the processor creates a new instance of an entry relationship type for the new list. At decision block 1430, the process determines whether the type of entry relationship being created is the same as the type of entry relationship defined in the original list. If yes, the

processing continues at process block 1440, where the processor copies the entry relationship's properties from original to new.

FIGURE 15 is a flow diagram illustrating the logic performed by a general-purpose computer system for adding items to static lists formed in accordance with the present invention. The process begins at predefined process block 1510 invoked in response to a user input to add an item to a static list. At processing block 1520, the processor creates a new instance of an entry relationship type for the list.

FIGURE 16 is a flow diagram illustrating the logic performed by a general-purpose computer system for removing items from static lists formed in accordance with the present invention. The process begins at predefined process block 1610 invoked in response to a user input to remove an item from a static list. At processing block 1620, the processor deletes the list entry relationship from the list.

FIGURE 17 is a flow diagram illustrating the logic performed by a general-purpose computer system for deleting items from static lists formed in accordance with the present invention. The process begins at predefined process block 1710 invoked in response to a user input to delete an item from a static list. At processing block 1720, the processor first determines all of the list entry relationships that exist where the deleting item is the target. Once completed, the processor deletes all list entry relationships from the list where the list item is the target item 104. At processing block 1730, the processor deletes the item itself.

FIGURE 18 is a block diagram overview of an implementation of static lists formed in accordance with the present invention using XML files. In a processing system 1102 (FIGURE 11) that uses a conventional file system, it may be preferable to implement static lists using an XML file 1810 to represent the non-holding references 1830 to the items 150 within the list 310. The non-holding references 1830 are those references that cannot be dynamically resolved should the item itself change location or be deleted. The XML file 1810 permits the processing system 1102 to advantageously serialize the links to the referenced items in the form of shell link data 1840. The shell link data 1840 is used in favor of any absolute path referring to the item 150 as it contains a persisted moniker to the referenced item. The shell link data 1840 also includes hints that permit the processing system 1102 to resolve the reference 1830 in cases where the target item has been moved. For example, the hints may include such things as item creation date and various forms of the file system path.

The XML file 1810 further permits the processing system 1102 to store and track user-defined arbitrary metadata 1820 to represent the properties 130 of the items 150 and the relationships 110. In such an implementation, the properties 130 are identified by their assigned globally unique identification (GUID) plus the property identification, also referred to in the Windows® operating system as the PROPERTYKEY. The metadata 1820 may also be advantageously employed to propagate certain security features for the static list to the referenced items 150.

FIGURE 19 is a block diagram overview of an implementation 1900 of static lists formed in accordance with the present invention using file system containers. In a processing system 1102 (FIGURE 11) employing a more advanced file system to manage data using a relational database, it is preferable to model the static list 320 as a file system container 1910. A file system container 1910 is a file object that includes holding references 1920 to items 150 as well as the relationships 110 between the items, depending on whether the referenced items are stored on the same volume as the container 1910.

In an example scenario, a user wants to produce a list of documents used to give presentations to clients about his company's new product, a brake pad. The documents include various Word® documents that describe the brake pad technology in depth, a PowerPoint® presentation, pictures of the brake pads, and even some video files shown the brake pads in action using an infrared camera. The user gives the presentation to different clients having different needs, cares, and wants. As a result, the user wishes to customize the presentation. Using static lists, the user can create different static lists, each with references to the same items, but in a different order (to tune the presentation to the audience). The user can also include different important properties. For example, for one client the sales price on all items is shown in the clear (and may even be specific to a client), whereas for other clients, the sales price is masked. In yet another example, the user may include properties that reveal the latest details of guarantees and awards they have won.

In the example scenario, the static lists are maintained automatically. When the user deletes one of the documents from one of the lists, the document is still available in all of the other lists where it is referenced. On the other hand, when the user deletes one of the documents from the folder where it resides, all lists referencing that document are updated to remove the reference so that the reference does not display as a dead link.

As a result of the foregoing, the user can advantageously create an unlimited number of static lists customized for a particular audience, and yet avoid the hassles of managing all of the references in those lists.

While the presently preferred embodiments of the invention have been illustrated and described, it will be appreciated that various changes may be made therein without departing from the spirit and scope of the invention. For example, it should be noted that either of the above-described implementations may be employed on a processing system 1102 regardless of what type of file system is employed. It may be advantageous to represent a static list as an XML file 1810, even on processing systems 1102 capable of using containers 1910, where interoperability with systems using more conventional file systems is desired. Moreover, in other embodiments, regardless of the type of file system employed, the items in the static list may be presented to the user using any user interface, including in a folder of the Windows® Shell user interface. As various operations are performed on the static list or the items in the list, the operations are either handled by the folder or delegated to a target of the referenced item, i.e., the target item.

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. A method for managing data in a list, the method comprising:

creating a list having an item type and a relationship type; and

adding an item to the list by generating an entry in the list in accordance with the relationship type, the entry representing an association between an item and the item type.

2. The method of Claim 1, further comprising changing a location of the item and updating the entry to refer to the changed location.

3. The method of Claim 1, wherein the status of the item changes when the item is deleted, and updating the entry includes removing the entry from the list.

4. The method of Claim 1, further comprising defining a property for the relationship type, wherein generating an entry in the list in accordance with the relationship type includes generating a value for the property.

5. The method of Claim 4, further comprising moving the item to a new list, wherein moving includes deleting the entry from the original list and generating an entry in the new list and copying the value for any property that the new list's relationship type has in common with the original list's relationship type.

6. The method of Claim 5, further comprising copying the item to a new list, wherein copying includes generating an entry in the new list and copying the value for any property that the new list's relationship type has in common with the original list's relationship type.

7. The method of Claim 1, the method further comprising deleting an item, wherein deleting includes removing the entry from the list and removing any other entry in other lists, where the other entry also represents an association with the item.

8. The method of Claim 1, the method further comprising applying a view to the list, wherein applying the view includes retrieving entries in the list having properties that match properties specified in the view and applying a display characteristic to the values of the matching properties.

9. The method of Claim 1, wherein the list is a file in XML format, wherein the entry is a non-holding reference to the item associated with the item type and the property is a metadata associated with the item, and updating the entry includes serializing a shell link to the reference.

10. The method of Claim 1, wherein the list is a file system container, and the entry is a holding reference to an item, the holding reference reflecting a current status of the item.

11. A system for managing data, the system comprising:

a storage medium for storing items of data and a list entry template;

a processing unit for operating a process to generate a list of selected items in response to a user input, wherein each entry of the list represents a reference to the item independent of the item's location in the storage medium, and wherein each entry includes a property value generated in accordance with the list entry template; and

a display unit for displaying a view of the items in the list, the view including a display of the property values of the entry in accordance with a display characteristic.

12. The system of Claim 11, wherein the stored items of data are moved to a new location and the process to generate the list of items includes a process to update the entry to refer to the new location.

13. The system of Claim 12, wherein the process to update the entry includes removing the entry from the list when the item is no longer stored on the storage medium.

14. The system of Claim 11, wherein the process to generate the list includes a process to copy the item to a new list comprising making an entry in the new list and copying the property value from the original entry to the new entry in accordance with the new list's entry template.

15. The system of Claim 13, wherein the process to generate the list includes a process to move the item to a new list including the process to copy the item to the new list plus a process to delete the entry from the original list.

16. The system of Claim 11, wherein the processing unit is to further operate a process to delete an item from the storage medium that includes removing all entries that refer to the item.

17. The system of Claim 11, wherein the generated list is a file in XML format, and wherein the entry in the list is a non-holding reference to the item and the property is a metadata associated with the item, and updating the entry includes serializing a shell link to the reference.

18. The system of Claim 11, wherein the generated list is a file system container, and the entry is a holding reference to the item, the holding reference referring to a current location of the item.

19. A computer-accessible medium having a computer-executable component for:

defining a list having an item type and a relationship type;

adding an item to the list by generating an entry in the list in accordance with the relationship type, the entry representing an association between an item and the item type; and

updating the entry whenever a status of the item changes.

20. The computer-accessible medium of Claim 19, wherein the computer-executable component updates the entry to refer to a current location of the item, regardless of an actual location of the item.

21. The computer-accessible medium of Claim 19, wherein the computer-executable component automatically removes the entry from the list when the item is deleted.

22. The computer-accessible medium of Claim 19, wherein the computer-executable component further defines a property for the relationship type, wherein generating an entry in the list in accordance with the relationship type includes generating a value for the property.

23. The computer-accessible medium of Claim 19, wherein the computer-executable component further moves the item to a new list, wherein moving includes

deleting the entry from the original list and generating an entry in the new list and copying the value for any property that the new list's relationship type has in common with the original list's relationship type.

24. The computer-accessible medium of Claim 19, wherein the computer-executable component further copies the item to a new list, wherein copying includes generating an entry in the new list and copying the value for any property that the new list's relationship type has in common with the original list's relationship type.

25. The computer-accessible medium of Claim 19, wherein the computer-executable component further applies a view to the list, wherein applying the view includes retrieving entries in the list having properties that match properties specified in the view and applying a display characteristic to the values of the matching properties.

26. The computer-accessible medium of Claim 19, wherein the list is stored in XML format, wherein the entry is a non-holding reference to the item associated with the item type and the property is a metadata associated with the item, and updating the entry includes serializing a shell link to the reference.

27. The computer-accessible medium of Claim 19, wherein the list is stored in a file system container, and the entry is a holding reference to an item, the holding reference reflecting a current status of the item.