EXHIBIT 9

US007945544B2

(12) **United States Patent**
Farber et al.

(10) **Patent No.: US 7,945,544 B2**
(45) **Date of Patent: May 17, 2011**

(54) **SIMILARITY-BASED ACCESS CONTROL OF DATA IN A DATA PROCESSING SYSTEM**

(75) Inventors: **David A. Farber**, Ojai, CA (US); **Ronald D. Lachman**, Northbrook, IL (US)

(73) Assignees: **Kinetech, Inc.**, Studio City, CA (US); **Level 3 Communications, LLC**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 624 days.

(21) Appl. No.: **11/980,688**

(22) Filed: **Oct. 31, 2007**

(65) **Prior Publication Data**

US 2008/0065635 A1 Mar. 13, 2008

Related U.S. Application Data

(60) Continuation of application No. 11/724,232, filed on Mar. 15, 2007, which is a continuation of application No. 11/017,650, filed on Dec. 22, 2004, which is a continuation of application No. 09/987,723, filed on Nov. 15, 2001, now Pat. No. 6,928,442, which is a continuation of application No. 09/283,160, filed on Apr. 1, 1999, now Pat. No. 6,415,280, which is a division of application No. 08/960,079, filed on Oct. 24, 1997, now Pat. No. 5,978,791, which is a continuation of application No. 08/425,160, filed on Apr. 11, 1995, now abandoned, application No. 11/980,688, which is a continuation of application No. 10/742,972, filed on Dec. 23, 2003, which is a division of application No. 09/987,723, which is a continuation of application No. 09/283,160, which is a division of application No. 08/960,079, which is a continuation of application No. 08/425,160.

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **707/698**; 707/821; 707/822

(58) **Field of Classification Search** 707/609, 707/821–828, 697–698
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,668,647 A 6/1972 Evangelisti et al.
3,835,260 A 9/1974 Prescher et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 0 268 069 A2 5/1988
(Continued)

OTHER PUBLICATIONS

Fowler, et al. "A User-Level Replicated File System," AT&T Bell Laboratories Technical Memorandum 0112670-930414-05, Apr. 1993, and USENIX 1993 Summer Conference Proceedings, Cincinnati, OH, Jun. 1993.

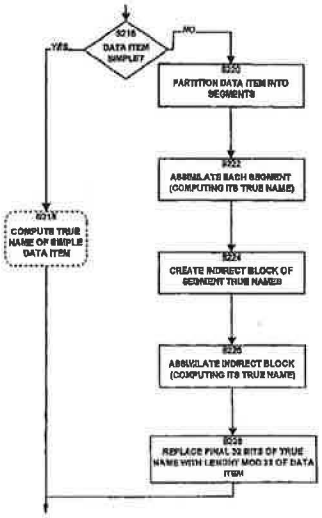
(Continued)

Primary Examiner — Khanh B Pham
(74) *Attorney, Agent, or Firm* — Davidson Berquist Jackson & Gowdey, LLP; Brian Siritzky

(57) **ABSTRACT**

Similarity of data items is determined by analyzing corresponding segments of the data items. A function is applied to each segment of a data item and the output of that function is compared to the output of the same function applied to a corresponding segment of another data item. A function may be applied to the output of the functions. The functions may be hash or message digest functions.

56 Claims, 31 Drawing Sheets



SIMILARITY-BASED ACCESS CONTROL OF DATA IN A DATA PROCESSING SYSTEM

RELATED APPLICATIONS

This application is a continuation of and claims priority to co-pending U.S. patent application Ser. No. 11/724,232 filed Mar. 15, 2007, which is a continuation of co-pending application Ser. No. 11/017,650, filed Dec. 22, 2004, which is a continuation of pending application Ser. No. 10/742,972, filed Dec. 23, 2003, which is a continuation of Ser. No. 09/987,723, filed Nov. 15, 2001, patented as U.S. Pat. No. 6,928,442; which is a which is a continuation of application Ser. No. 09/283,160, filed Apr. 1, 1999, now U.S. Pat. No. 6,415,280, which is a division of application Ser. No. 08/960,079, filed Oct. 24, 1997, now U.S. Pat. No. 5,978,791, which is a continuation of Ser. No. 08/425,160, filed Apr. 11, 1995, now abandoned, the contents of which each of these applications are hereby incorporated herein by reference. This application is a continuation of and claims priority to co-pending application Ser. No. 11/017,650, filed Dec. 22, 2004, which is a continuation of application Ser. No. 09/987,723, filed Nov. 15, 2001, now U.S. Pat. No. 6,928,442, which is a continuation of application Ser. No. 09/283,160, filed Apr. 1, 1999, now U.S. Pat. No. 6,415,280, which is a division of application Ser. No. 08/960,079, filed Oct. 24, 1997, now U.S. Pat. No. 5,978,791, which is a continuation of Ser. No. 08/425,160, filed Apr. 11, 1995, now abandoned, the contents of which each of these applications are hereby incorporated herein by reference. This is also a continuation of and claims priority to co-pending application Ser. No. 10/742,972, filed Dec. 23, 2003, which is a division of application Ser. No. 09/987,723, filed Nov. 15, 2001, now U.S. Pat. No. 6,928,442, which is a continuation of application Ser. No. 09/283,160, filed Apr. 1, 1999, now U.S. Pat. No. 6,415,280, which is a division of application Ser. No. 08/960,079, filed Oct. 24, 1997, now U.S. Pat. No. 5,978,791, which is a continuation of Ser. No. 08/425,160, filed Apr. 11, 1995, now abandoned, the contents of which each of these applications are hereby incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to data processing systems and, more particularly, to data processing systems wherein data items are identified by substantially unique identifiers which depend on all of the data in the data items and only on the data in the data items.

2. Background of the Invention

Data processing (DP) systems, computers, networks of computers, or the like, typically offer users and programs various ways to identify the data in the systems.

Users typically identify data in the data processing system by giving the data some form of name. For example, a typical operating system (OS) on a computer provides a file system in which data items are named by alphanumeric identifiers. Programs typically identify data in the data processing system using a location or address. For example, a program may identify a record in a file or database by using a record number which serves to locate that record.

In all but the most primitive operating systems, users and programs are able to create and use collections of named data items, these collections themselves being named by identifiers. These named collections can then, themselves, be made part of other named collections. For example, an OS may provide mechanisms to group files (data items) into directories (collections). These directories can then, themselves be made part of other directories. A data item may thus be identified relative to these nested directories using a sequence of names, or a so-called pathname, which defines a path through the directories to a particular data item (file or directory).

As another example, a database management system may group data records (data items) into tables and then group these tables into database files (collections). The complete address of any data record can then be specified using the database file name, the table name, and the record number of that data record.

Other examples of identifying data items include: identifying files in a network file system, identifying objects in an object-oriented database, identifying images in an image database, and identifying articles in a text database.

In general, the terms "data" and "data item" as used herein refer to sequences of bits. Thus a data item may be the contents of a file, a portion of a file, a page in memory, an object in an object-oriented program, a digital message, a digital scanned image, a part of a video or audio signal, or any other entity which can be represented by a sequence of bits. The term "data processing" herein refers to the processing of data items, and is sometimes dependent on the type of data item being processed. For example, a data processor for a digital image may differ from a data processor for an audio signal.

In all of the prior data processing systems the names or identifiers provided to identify data items (the data items being files, directories, records in the database, objects in object-oriented programming, locations in memory or on a physical device, or the like) are always defined relative to a specific context. For instance, the file identified by a particular file name can only be determined when the directory containing the file (the context) is known. The file identified by a pathname can be determined only when the file system (context) is known. Similarly, the addresses in a process address space, the keys in a database table, or domain names on a global computer network such as the Internet are meaningful only because they are specified relative to a context.

In prior art systems for identifying data items there is no direct relationship between the data names and the data item. The same data name in two different contexts may refer to different data items, and two different data names in the same context may refer to the same data item.

In addition, because there is no correlation between a data name and the data it refers to, there is no a priori way to confirm that a given data item is in fact the one named by a data name. For instance, in a DP system, if one processor requests that another processor deliver a data item with a given data name, the requesting processor cannot, in general, verify that the data delivered is the correct data (given only the name). Therefore it may require further processing, typically on the part of the requestor, to verify that the data item it has obtained is, in fact, the item it requested.

A common operation in a DP system is adding a new data item to the system. When a new data item is added to the system, a name can be assigned to it only by updating the context in which names are defined. Thus such systems require a centralized mechanism for the management of names. Such a mechanism is required even in a multi-processing system when data items are created and identified at separate processors in distinct locations, and in which there is no other need for communication when data items are added.

In many data processing systems or environments, data items are transferred between different locations in the system. These locations may be processors in the data processing system, storage devices, memory, or the like. For example,

exact contents of the directory tree at the time it was frozen. The frozen directory can be copied with its components preserved.

The Acquire True File remote mechanism (used in mirroring and archiving) preserves the directory tree structure by ensuring that all of the component segments and True Files in a compound data item are actually copied to a remote system. Of course, no transfer is necessary for data items already in the registry of the remote system.

In operation, the system can efficiently make a copy of any collection of data items, to support a version control mechanism for groups of the data items.

The Freeze Directory primitive mechanism is used to create a collection of data items. The constituent files and segments referred to by the frozen directory are maintained in the registry, without any need to make copies of the constituents each time the directory is frozen.

Whenever a pathname is traversed, the Get Files in Directory operating system mechanism is used, and when it encounters a frozen directory, it uses the Expand Frozen Directory primitive mechanism.

A frozen directory can be copied from one pathname to another efficiently, merely by copying its True Name. The Copy File operating system mechanism is used to copy a frozen directory.

Thus it is possible to efficiently create copies of different versions of a directory, thereby creating a record of its history (hence a version control system).

In operation, the system can maintain a local inventory of all the data items located on a given removable medium, such as a diskette or CD-ROM. The inventory is independent of other properties of the data items such as their name, location, and date of creation.

The Inventory Existing Directory extended mechanism provides a way to create True File Registry entries for all of the files in a directory. One use of this inventory is as a way to pre-load a True File registry with backup record information. Those files in the registry (such as previously installed software) which are on the volumes inventoried need not be backed up onto other volumes.

The Inventory Removable, Read-only Files extended mechanism not only determines the True Names for the files on the medium, but also records directory entries for each file in a frozen directory structure. By copying and modifying this directory, it is possible to create an on line patch, or small modification of an existing read-only file. For example, it is possible to create an online representation of a modified CD-ROM, such that the unmodified files are actually on the CD-ROM, and only the modified files are online.

In operation, the system tracks possession of specific data items according to content by owner, independent of the name, date, or other properties of the data item, and tracks the uses of specific data items and files by content for accounting purposes. Using the Track for Accounting Purposes extended mechanism provides a way to know reliably which files have been stored on a system or transmitted from one system to another.

True Names in Relational and Object-Oriented Databases

Although the preferred embodiment of this invention has been presented in the context of a file system, the invention of True Names would be equally valuable in a relational or object-oriented database. A relational or object-oriented database system using True Names would have similar benefits to those of the file system employing the invention. For instance, such a database would permit efficient elimination of duplicate records, support a cache for records, simplify the process of maintaining cache consistency, provide location-independent access to records, maintain archives and histories of records, and synchronize with distant or disconnected systems or databases.

The mechanisms described above can be easily modified to serve in such a database environment. The True Name registry would be used as a repository of database records. All references to records would be via the True Name of the record. (The Local Directory Extensions table is an example of a primary index that uses the True Name as the unique identifier of the desired records.)

In such a database, the operations of inserting, updating, and deleting records would be implemented by first assimilating records into the registry, and then updating a primary key index to map the key of the record to its contents by using the True Name as a pointer to the contents.

The mechanisms described in the preferred embodiment, or similar mechanisms, would be employed in such a system. These mechanisms could include, for example, the mechanisms for calculating true names, assimilating, locating, realizing, deleting, copying, and moving True Files, for mirroring True Files, for maintaining a cache of True Files, for grooming True Files, and other mechanisms based on the use of substantially unique identifiers.

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiments, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

We claim:

1. A computer-implemented method, the method comprising:

(A) for a first data item comprising a first plurality of parts,

(a1) applying a first function to each part of said first plurality of parts to obtain a corresponding part value for each part of said first plurality of parts, wherein each part of said first plurality of parts comprises a corresponding sequence of bits, and wherein the part value for each particular part of said first plurality of parts is based, at least in part, on the corresponding bits in the particular part, and wherein two identical parts will have the same part value as determined using said first function, wherein said first function comprises a first hash function; and

(a2) obtaining a first value for the first data item, said first value obtained by applying a second function to the part values of said first plurality of parts of said first data item, said second function comprising a second hash function;

(B) for a second data item comprising a second plurality of parts,

(b1) applying said first function to each part of said second plurality of parts to obtain a corresponding part value for each part of said second plurality of parts, wherein each part of said second plurality of parts consists of a corresponding sequence of bits, and wherein the part value for each particular part of said second plurality of parts is based, at least in part, on the corresponding bits in the particular part of the second plurality of parts; and

(b2) obtaining a second value for the second data item by applying said second function to the part values of said second plurality of parts of said second data item; and

(C) ascertaining whether or not said first data item corresponds to said second data item based, at least in part, on said first value and said second value.

2. The method of claim **1** wherein said first data item corresponds to said second data item when said first value is identical to said second value.

3. The method of claim **1** wherein said ascertaining in (C) is used to determine whether the first data item matches the second data item.

4. The method of claim **1** wherein said ascertaining in (C) is used to determine whether the first data item is a copy of the second data item.

5. The method of claim **1** wherein the first function is the same as the second function.

6. The method of claim **1** wherein the first plurality of parts of the first data item are non-overlapping, and wherein the second plurality of parts of the second data item are non-overlapping.

7. The method of claim **1** wherein each of the parts in the first plurality of parts and each of the parts in the second plurality of parts is the same size.

8. A computer-implemented method comprising:

(A) maintaining a database of values, at least one value for each data item of a plurality of data items,

wherein each data item of the plurality of data items comprises a corresponding one or more parts, and

wherein each of the one or more parts of each data item comprises a corresponding sequence of bits, and

wherein each of the one or more parts of each data item has a corresponding part value, the part value for each particular part being based on a first given function of the corresponding sequence bits for that particular part,

wherein two identical parts will have the same part value as determined using the first given function, and

the value for each particular data item being based, at least in part, on a second given function of the part values of the one or more parts of that particular data item, and

wherein the first given function comprises a first hash function, and the second given function comprises a second hash function;

(B) obtaining a second value, the second value corresponding to a second data item, the second data item comprising a corresponding one or more parts,

each of the one or more parts of the second data item comprising a corresponding sequence of bits,

each of the one or more parts of the second data item having a corresponding part value,

wherein the part value for each particular part of the second data item is based on the first given function of the corresponding sequence of bits in that particular part of the second data item; and

wherein the second value is based on the second function of the one or more part values of the second data item; and

(C) ascertaining whether or not the second data item corresponds to any of the plurality of data items, based, at least in part, on whether or not the second value corresponds to any value in the database of values.

9. The method of claim **8** wherein the second value corresponds to a particular value in the database of values when the second value is equal to the particular value in the database of values.

10. The method of claim **8** wherein the first hash function is selected from the functions MD5 and SHA.

11. The method of claim **10** wherein the second hash function is selected from the functions MD5 and SHA.

12. The method of claim **8** wherein the first given function is the same as the second given function.

13. The method of claim **8** wherein the database comprises a mapping from data item values to corresponding data items.

14. The method of claim **8** wherein the second value is obtained as part of a search for the second data item.

15. The method of claim **8** wherein the second value is obtained as part of a search for data items matching the second data item.

16. The method of claim **8** further comprising:

(D) when the second value corresponds to a particular value in the database, providing information about a particular data item corresponding to the particular entry.

17. The method of claim **8** wherein the step (B) of obtaining the second value comprises calculating the second value.

18. The method of claim **8** wherein at least some of the data items are files.

19. The method of claim **8** wherein the database comprises a mapping from values to corresponding data items, and wherein, when the second value corresponds to the first value, information about the corresponding data item is provided.

20. The method of claim **8** wherein the parts are segments.

21. The method of claim **8** further comprising:

(D) adding a new value to the database, wherein the new value corresponds to a new data item distinct from the plurality of data items.

22. A computer-implemented method comprising:

(A) obtaining a particular data item value corresponding to a particular data item, the particular data item comprising a corresponding one or more parts,

each of the one or more parts of the particular data item comprising a corresponding sequence of bits,

each of the one or more parts of the particular data item having a corresponding part value,

wherein the part value for each specific part of the one or more parts of the particular data item is based, at least in part, on a first given function of the corresponding sequence of bits in that specific part of the particular data item;

wherein two identical parts will have the same part value as determined using the first given function, and

wherein the particular data item value is based, at least in part, on a second given function of the one or more part values of the particular data item,

wherein the first given function comprises a first hash function, and the second given function comprises a second hash function; and

(B) ascertaining whether or not the particular data item corresponds to any of a plurality of data items, based, at least in part, on whether or not the particular data item value corresponds to any value in a database of data item values,

wherein the database of data item values comprises at least one data item value for each data item of the plurality of data items,

wherein each data item of the plurality of data items comprises a corresponding one or more parts, and

wherein each of the one or more parts of each data item of the plurality of data items comprises a corresponding sequence of bits, and

wherein each of the one or more parts of each data item of the plurality of data items has a corresponding part value, the part value for each particular part of the one or

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.