PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 A1

(11) International Publication Number: WO 98/50866

(43) International Publication Date: 12 November 1998 (12.11.98)

(21) International Application Number: PCT/NO98/00139

(22) International Filing Date: 6 May 1998 (06.05.98)

(30) Priority Data:
08/852,099 6 May 1997 (06.05.97) US

(71) Applicant: BIRDSTEP TECHNOLOGY AS [NO/NO]; Haakon VIIs gate 6, N-0161 Oslo (NO).

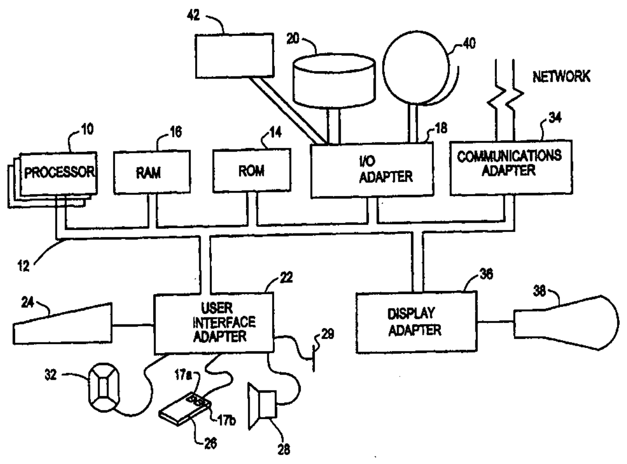
(72) Inventor: VETHE, Olaf; Gamlelinja 49B, N-1254 Oslo (NO).

(74) Agent: OSLO PATENTKONTOR AS; Postboks 7007 M, N-0306 Oslo (NO).

(81) Designated States: AU, CN, HU, JP, KR, NZ, PL, SG, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEM AND METHOD FOR STORING AND MANIPULATING DATA IN AN INFORMATION HANDLING SYSTEM



(57) Abstract

The present invention is directed to a database and database management system and method designed to store and manipulate any type of data and any combination of data types. The underlying data architecture is uniquely flexible, and thus to the DBMS, the data in the database will appear to be of a type consistent with the access method of the DBMS. Further, the present invention allows a number of simultaneous, different access methods to the same underlying data, delivering the ability to work with complex data within one easily managed system. One of the key aspects of the present invention is the "atomization" of the data. Atomization is the separation of the storage of data content from its definition, as well as from its occurrences or instances in the database. An atom is the most basic element in the database. Data content is stored in content atoms, data definitions are stored in type atoms, and each instance of the same data value/property is represented by instance atoms. When connected, the three different atom types form a molecule. Complex data may be represented in the database by linking together instance atoms from several molecules to form inner relations, and then by further linking together inner relations to form outer relations.

SYSTEM AND METHOD FOR STORING AND MANIPULATING DATA IN AN INFORMATION HANDLING SYSTEM

Field of the Invention

The present invention relates to information handling systems, and, more particularly, to a system and method for storing and manipulating data in an information handling system.

Background of the Invention

A database consists of one or more large sets of persistent data. Typically, users can update and query the database using software associated with the database.

A database is the data stored by a database management system (DBMS). A DBMS is a set of software programs that control the organization, storage, and retrieval of data in a database. A DBMS also controls the security and integrity of the database. Typically, a DBMS also provides an interactive query facility, which allows a user to interactively search and analyze data from the database.

There are several prior art methods available for organizing data in a database. The three most common types of prior art databases are hierarchical, network, and relational. A DBMS may provide one or more of these, and other, types of database organization.

In a hierarchical database, the data items are referred to as records, and are stored in a tree structure. Hierarchical databases link records together in a manner similar to a typical organization chart. This means that each record can be owned by only one owner record. For example, a department record may "own" fifteen employee records. However, each employee record may only be owned by one owner record, in this case by its department record. This makes it difficult to model real world situations using a hierarchical database. For example, an employee may be both a member of a department and a member of a team made up of employees from several departments. However, a hierarchical database would not allow the same employee record to be owned by both a department record and a team record.

A network database is similar to a hierarchical database, however, data records may be freely interconnected, with no requirement that the data records fit in a tree structure. In a network database, an employee record could be owned by both a department record and a team record.

Both hierarchical databases and network databases are time-consuming to search and difficult to change. Changing the data structures in a hierarchical or a network database typically requires shutting down the database and rebuilding it.

Another type of prior art database is a relational database. In a relational database, all data is stored in simple tables, referred to as relations. Relational databases remove the complex relationships between records found in hierarchical and network databases. The design of the records in a relational database provides a common field, such as employee number, for matching. Often, the fields used for matching are indexed in order to speed up searching.

However, there are several disadvantages to relational databases. Relational databases are complex and unnatural for many data structures (i.e. network type data structures). Relational databases are redundant, as many fields are stored in more than one relation. While the use of index fields can increase query speed, the space needed to store the indexes can sometimes become significantly larger than the space needed to store the data in the database. The use of indexes is also redundant. This redundancy, along with the redundant storage of data fields in more than one relation, can cause performance degradation when there are a high volume of updates to the database. Finally, the administration cost of a relational database is high, as data must be frequently reorganized to keep performance acceptable.

Because of the many disadvantages of prior art databases, such as hierarchical, network, and relational models, some software manufacturers have begun developing object-oriented databases (OODBs). However, the OODBs that exist today use traditional storage technology (i.e. relational and other) to actually store the data. Current OODBs and object database management systems (ODMSs) are really object-oriented interfaces to old database technologies and old database management systems. Current OODBs and ODMSs actually try, although not very successfully, to use today's currently existing technology (i.e. relational and other) to store object-oriented data.

Consequently, it would be desirable to have a database, and database management system and method, for organizing data so that it can be accessed as if it were any conceivable database organization, including those discussed above. It would be desirable if the system and method could support many simultaneous, different access methods and storage/retrieval syntaxes. Further, it would be desirable if the system and method allowed fast and efficient searching, dynamic schema evolution with no need to take the database off-line for restructuring, and automatic history generation.

Brief Summary of the Invention

Accordingly, the present invention is directed to a database and database management system and method designed to store and manipulate any type of data (i.e. text, numeric, spatial, graphical, etc.) and any combination of data types. The underlying data architecture is uniquely flexible, and thus to the DBMS, the data in the database will appear to be of a type consistent with the access method of the DBMS (although, of course, the underlying structure of the database does not change). Further, the present invention allows a number of simultaneous, different access methods to the same underlying data, delivering the ability to work with complex data within one easily managed system.

One of the key aspects of the present invention is the "atomization" of the data. Atomization is the separation of the storage of data content from its definition, as well as from its occurrences or instances in the database. An atom is the most basic element in the database. Data content is stored in content atoms, data definitions are stored in type atoms, and each instance of the same data value/property is represented by instance atoms. When connected, the three different atom types form a molecule. Complex data may be represented in the database by linking together instance atoms from several molecules to form inner relations, and then by further linking together inner relations to form outer relations.

One advantage of the present invention is that it is able to automatically maintain a separate chronological history for every instance atom in the database. When modifying the content atom (i.e. the data value) of a molecule, the database will (at the user's option) either reconnect the instance atom to a new content atom, or generate a new instance with the new content and link the old instance into a history chain.

Another advantage of the present invention is a unique search structure which allows for fast and efficient searching of the database. For each content atom, the DBMS uses the 'n' most significant bytes in the data as a vector into the system's search structure. The DBMS splits the search structure into 'm' separate mini-structures where 'm' is the range of the 'n' most significant bytes. The actual content atom is then stored in its vector's mini-structure.

Brief Description of the Drawings

The foregoing and other features and advantages of the present invention will become more apparent from the detailed description of the best mode for carrying out the invention as rendered below. In the description to follow, reference will be made to the accompanying drawings, where like reference numerals are used to identify like parts in the various views and in which:

Figure 1 is a block diagram of an information handling system capable of storing and manipulating the atomic database of the present invention;

Figure 2 depicts a content atom, type atom, and instance atom linked together to form a molecule;

Figure 3 depicts several molecules linked together to form an inner relation;

Figure 4 depicts two inner relations linked together to form an outer relation;

Figure 5 illustrates the manner in which history links may be used to maintain instance history;

Figure 6 depicts the internal components of the database of the present invention;

Figure 7 depicts further details regarding the internal structure of the database of the present invention;

Figure 8 depicts an internal structure of the database used to improve searching performance;

Figure 9 depicts the search structure of the present invention;

Figure 10 depicts the exposed access methods of the present invention;

Figure 11 is a flow chart illustrating a method of creating a database in accordance with the present invention;

Figure 12 is a flow chart illustrating a method of adding a content/instance atom to the database;

Figure 13 is a flow chart illustrating a method of searching the database; and

Figure 14 is a flow chart illustrating a method of updating a content/instance atom.

Detailed Description of the Preferred Embodiment

The invention may be implemented on a variety of hardware platforms, including personal computers, workstations, mini-computers, and mainframe computers. Many of the steps of the method of the present invention may be advantageously implemented on parallel processors of various types. Referring now to Figure 1, a typical configuration of an information handling system that may be used to practice the novel method of the present invention will be described. The computer system of Figure 1 has at least one processor 10. Processor 10 is interconnected via system bus 12 to random access memory (RAM) 16, read only memory (ROM) 14, and input/output (I/O) adapter 18 for connecting peripheral devices such as disk units 20, tape drives 40, and printers 42 to bus 12, user interface adapter 22 for connecting keyboard 24, mouse 26 having buttons 17a and 17b, speaker 28, microphone 32, and/or other user interface devices such as a touch screen device 29 to bus 12, communication adapter 34 for connecting the information handling system to a data processing network, and display adapter 36 for connecting bus 12 to display device 38. Communication adaptor 34 may link the system depicted in Figure 1 with hundreds or even thousands of similar systems, or other devices, such as remote printers, remote servers, or remote storage units.

The system and method of the present invention are designed to store and manipulate any type of data (i.e. text, numeric, spatial, graphical, etc.) and any combination of data types. The underlying data architecture is uniquely flexible. As a result, the database management system (DBMS) of the present invention can be a relational DBMS, an object DBMS, a hierarchical DBMS, or any other DBMS. In addition, each structure's storage/retrieval syntax (e.g. SQL for relational, OQL for object, etc.) may be used to access and retrieve data from the database. To the DBMS, the data in the database will appear to be of a type consistent with the access method of the DBMS (although, of course, the underlying structure of the database does not change). Further, the present invention allows a number of simultaneous, different access methods to the same underlying data, delivering the ability to work with complex data within one easily managed system.

One of the key aspects of the present invention is the "atomization" of the data. Atomization is the separation of the storage of data content (e.g. the string "John Smith") from its definition (i.e. its data type and relationship to other data elements), as well as from its occurrences or instances in the database.

An "atom," also referred to as an "element," is the most basic element in the database. Referring now to Figure 2, the relationship among the three elements, or atom types, is depicted. Data content, in this case "John Smith," is stored in content atom 50, its definition is stored in type atom 52, and each instance of the same data value/property is represented by instance atom 54. When connected, the three different atom types 50, 52, 54 form molecule 56. Type atoms 52 and instance atoms 54 consist of a data structure of values and pointers. The DBMS manages these atoms and the many possible relationships between them.

Type atom 52 contains the definition of instance atom 54 (referred to as a field definition in prior art terminology). This definition is a data structure with a number of elements including both a typename and a basetype. The typename could be "Name," "Birthdate," "Patient," "Supplier," etc. The basetype determines how the data is physically represented in the database (i.e. name = text, birthdate = long integer, etc.).

Content atom 50 holds data values (i.e, "John Smith," "10091962," etc.). Each data value is stored only once in the database. Multiple instances of the same data value are differentiated by their instance atoms 54 (as described below with reference to Figures 6 and 8). Even if a data value is common to several type atoms 52 it is still physically stored just once. For example, "John Smith" could be an instance of type atoms "Attorney," "Parent," and "Supplier," but only one content atom with "John Smith" as its data value is created. Content atom 50 holds both the data value and a pointer to the structure of its instance atoms 54 (as described below with reference to Figures 6 and 8).

Instance atom 54 acts as the connector between type atom 52 and content atom 50. Each instance atom 54 defines one occurrence of a combination (i.e., Type atom "Attorney" + Content atom "John Smith"), which represents an instance of a field or object in the database.

Molecule 56 is depicted in Figure 2, and consists of one atom of each type 50, 52 bound through an instance atom 54. An incomplete molecule (not shown) consists of an instance atom 54 connected to either a type atom 52 or a content atom 50, but not to both. An incomplete molecule may be used to store data with no predefined type. This allows the flexibility to store data, and decide on its type later. An incomplete molecule may also be used to define instances with no data or with undefined data. This adds a flexibility to database design that is not available in the prior art, but is often needed. For example, suppose the DBMS receives unstructured data from optical scanning of newspaper advertisements. It is impossible for the DBMS to know the type of the various text items in an advertisement, because the placement of different items (phone, fax, address, name, description etc.) will be different for each advertisement. Therefore, all items initially do not have a type, but because their content is known a user may search for the items. At a later point in time, specific types may be allocated to the items.

Referring now to Figure 3, an inner relation 60 is depicted. In this example, inner relation 60 consists of instance atoms 54, 62, 64, and 66, which are linked together with pointers 68. Inner relation 60 can be used as the equivalent to the data fields in a conventional record, or can be a more complex data type, such as a spatial data type, with one atom for each of "n" coordinates.

As shown in Figure 3, an owner is the first instance atom, in this case instance atom 54, in inner relation 60. The members are the instance atoms, including the owner, which belong to the inner relation. In the example shown in Figure 3, instance atoms 62, 64, and 66, along with owner atom 54, are the members of inner relation 60.

Inner relations may be linked together to form outer relations. One of the available basetypes is "pointer," which means that a content atom in a molecule can act as a pointer to a specific place in the database, usually an instance atom (e.g, an instance atom in another inner relation). This capability allows linking between distinct inner relations to form infinitely complex data structures.

Referring now to Figure 4, an example of an outer relation is illustrated. In Figure 4, there are two inner relations shown, inner relation Person-1 80 and inner relation Person-2 82. Person-1 80 consists of three instance atoms 54, 82, and 84.

Instance atom 54 is linked to content atom 50 and type atom 52. Instance atom 82 is linked to content atom 86 and type atom 88. Instance atom 84 is linked to content atom 90 and type atom 92. Person-2 82 consists of two instance atoms 94 and 96. Instance atom 94 is linked to content atom 98 and type atom 100. Instance atom 96 is linked to content atom 102 and type atom 104.

In the example depicted in Figure 4, Person-1 80 is the parent of Person-2 82 (note that type atom 92 is of type "CHILD"). Content atom 90 contains the address of instance atom 94 (as denoted by dashed line 106). Person-1 80 could be linked to several children through the use of additional address links, similar to the address link of content atom 90. The combination of two or more inner relations forms an outer relation, which in this case could represent an entire family.

The system and method of the present invention is able to automatically maintain a separate chronological history for every instance atom in the database. When modifying the content atom (i.e. the data value) of a molecule, the database will (at the user's option) either reconnect the instance atom to a new content atom, or generate a new instance with the new content and link the old instance into a history chain. As a result, the database can automatically maintain a separate chronological history on every instance in the database.

An example of instance history formation is depicted in Figure 5. Type atom 110 is linked to three instance atoms, Instance1 112 (which is linked to Content1 114), Instance2 116 (which is linked to Content2 118), and Instance3 120 (which is linked to Content3 122). Note that history links 124 connect Instance3 120 with two older versions 126, 130 of Instance3 (along with their content atoms 128, 132).

The main internal components of the database of the present invention are shown in Figure 6. The components depicted in Figure 6 are stored in RAM 16 of Figure 1. The solid lines in Figure 6 show the links defining molecules, while the dashed lines show the links defining inner relations. As discussed above, withe reference to Figure 2, each molecule consists of a content atom 150, an instance atom 152, and a type atom 154. Inner relations are formed by defining links between instance atoms 152. Search structure 156 is defined in more detail below, with reference to Figure 9. Type atoms 154 are stored in dictionary 158, which is itself, a database according to the present invention, thus providing user-extensibility for type definitions.

Further details regarding the internal structure of the DBMS are illustrated in Figure 7. When each type atom 160 is created, it contains a set of characteristics. In the described embodiment, these characteristics include basetype 162, typename 164, type description 166, type handle 168, type low value 170, type high value 172, and type count (instance) 174. Each type atom 160 is allocated a unique type handle 168, which is a number that uniquely identifies the type atom 160. Type handle 168 is part of the information contained in each instance atom 176, thus uniquely identifying each instance atom 176 with its type atom 160.

Content atoms are maintained as an integral part of the database. Each content atom is part of a separate search structure that starts from the content's vector (described below with reference to Figure 9). Each instance atom is stored under its content atom in a two-level hierarchy, as depicted in Figure 8. Referring now to Figure 8, the instance atoms 200, 202, 204, 206 are shown at the lowest level of the hierarchy, grouped by type handle. At the first level below content atom 208, there are a set of "branches" 210, 212, 214 each of which contains a type handle. There is one branch for each of the type handles for which there is one or more corresponding instance atoms (and no branch if there are no instances of a particular type handle). These branches are linked in ascending order. This structure is used internally by the DBMS (i.e. it is not available to the user) for searches and for improving performance.

When inserting or updating data elements, the user (typically, an application programmer) locates the correct position in an inner relation by traversing the linking between instances. Next, the position is found under the content/branch hierarchy, and then the new instance (or modified instance) is stored and linked into both the instance chain and the content structure.

The system and method of the present invention includes a unique search structure which allows for fast and efficient searching of the database. For each content atom, the DBMS uses the 'n' most significant bytes in the data as a vector into the system's search structure. The DBMS splits the search structure into 'm' separate mini-structures where 'm' is the range of the 'n' most significant bytes. The actual content atom is then stored in its vector's mini-structure. The search structure is similar to a hash table except that no algorithm is needed to define its elements, and the elements are always in sort order. To enhance performance, there is a separate search structure for each basetype.

An example of a search is shown in Figure 9. Referring now to Figure 9, suppose that a user wishes to locate a particular instance of "John Smith" 220. In the described embodiment, the most significant byte (i.e. "J") is used as a vector into search structure 222. The entire content (i.e. "John Smith") is used to find the correct position within search structure 222. Only one content atom 224 in the entire databse contains the content of "John Smith." Once this content atom 224 is located, the user can then search all the instance atoms linked to the content atom to find the desired instance of "John Smith."

The method and system of the present invention also incorporate a variety of exposed access methods which are the sole means of access to the internal components. The exposed functions provide means for data storage and retrieval, defining and manipulating inner and outer relations and type atoms, as well as means for performing a variety of system functions. The exposed access methods are illustrated in Figure 10.

Referring now to Figure 10, the exposed access methods may be an developer access method 230 (perhaps in the form of a class library) for direct use by application developers. Users may also access the system through other implementations of this layer/interface which present the system as an SQL access method 232, object access method 234, network access method 236, or other database access method, such as a TCP/IP serialized HTML 238. Note that the underlying structure of atoms and links remains the same in all implementations and that data can be simultaneously accessed by different methods according to users' needs. Note also that the exposed access methods, as well as the database itself, is stored in RAM 16 of Figure 1. The access methods 230, 232, 234, 236, 238 interface to the database through the use of an application programmer interface (API) 240. In addition to the access methods shown, users may, through the use of other application programs (not shown), use API 240 to create a new database (as discussed below with reference to Figure 11).

Referring now to Figures 11 through 14, methods of using the present invention will be described. Figure 11 illustrates a method of creating a new database. Through the use of API 240, a user or user application program first creates a dictionary of type atoms (step 300). As discussed above with reference to Figure 6, the dictionary is also a database according to the present invention. Next, instance atoms and content atoms are created to represent the various data items stored in the database (step 302). Note that to a user, the combination of one instance atom and one content atom will usually be thought of as a single data item. Finally, the appropriate instance, content, and type atoms are linked together, through the use of pointers, to create molecules (step 304). To represent more complex data types, molecules may be linked together to form inner relations (as discussed above with reference to Figure 3) and inner relations may be linked together to form outer relations (as discussed above with reference to Figure 4).

Referring now to Figure 12, a method for adding new data to the database is illustrated. Data is received from the user (step 400) and the database is searched (step 401). Searching is described more fully below with reference to Figure 13. The system determines if the data content exists in the database (step 402). If not, a content atom is instantiated and linked into the search structure (step 403). If the data content does exist already, the system determines if there is more than one branch (step 404). A branch is a link from a content atom to one or more instance atoms of the same type. If the content atom is only associated with one type atom, a new instance is stored in the single branch (step 405). If the content atom is associated with more than one type atom, the system searches for the correct type and stores the new instance in the correct branch (step 406).

Referring now to Figure 13, a method for searching the database will now be described. Data is received from the user (step 500). The first "n" bytes of the data is used to select the proper mini-structure in which to begin the search (step 501). The mini-structure is then searched (step 502). The system determines if the data has been found (step 503). If not, the user is informed that the data does not currently exist in the database (step 504). If the data is found, it is output to the user (step 505).

Referring now to Figure 14, a method for updating a data item in the database will now be described. The system first finds the content/instance pair which is being changed (step 600). The new data is created (step 601). If history has been requested (step 602), a new instance atom is instantiated under the newly created data (step 603) and the new instance is set up to point to the old instance (step 604). If history has not been requested, the old instance becomes the new instance (step 605) and is relinked to the newly created data (step 606).

The unique architecture of the present invention provides many advantages over the prior art. The present invention provides dynamic schema evolution, eliminating the need to take the database off-line for restructuring and thus also eliminating the overhead associated with of on-line restructuring. The system is extensible, supporting new, user-defined data types. The database stores data codexes just once (e.g. the string "John Smith" is physically stored just once no matter how many John Smith's may occur in the database). This feature reduces the size of any database and, when combined with its search structure of the present invention (described above with reference to Figure 9), provides very high speed retrieval and update operations. The DBMS of the present invention eliminates the need for separately defined and maintained indexes because it maintains its own internal search structure for all data elements. The system is, therefore, more robust and requires less system administration.

The present invention automatically maintains a history of user-designated data elements, dramatically reducing the amount of programming effort needed to provide this facility in prior art systems. Maintaining history in prior art systems requires extensive programming and database design. In relational databases this is accomplished by defining separate history tables for each table where history is required. The system and method of the present invention provide this functionality automatically at the instance atom level with no separate external definitions by the developer or database designer. The invention's programming interface includes functions for reversing instance modifications using the history atoms. Programming for transaction reversal is minimized and conditions for system-initiated rollbacks can be easily defined.

The invention further supports object inheritance, encapsulation and polymorphism. The basetype of a type atom may change over time (through overloading) and a complete history is automatically maintained, eliminating the need to convert old data values as new definitions are added. For example, type atom "Temperature," may be defined as INTEGER (fixed number), but there may also be type atoms with typename "Temperature," with basetypes TEXT (formula) and BLOB (applet). As a result, data values can be accepted and stored in multiple formats. This functionality can be used to provide both inheritance and polymorphism. As another example, the typename SPEED can be defined as an integer (e.g, basic speed = 65 mph), text describing the speed (e.g. "very fast," "extremely slow," "faster than light"), and a BLOB (with an applet that animates a speed bar). Note that polymorphism can also be implemented through the different user/developer interfaces into the database.

A further advantage of the present invention is its universal object-relational-hierarchical capability. Each inner relation may consist of any combination of molecules (and therefore any combination of data types, number of instances, etc.) and each inner relation may be changed at will. The present invention therefore dispenses with the need to work with static and inflexible record and database structures. Complex data structures can be implemented as a "native" part of the database, instead of requiring complicated manipulation and transformation to fit into the database structure.

Data structures can be defined to match any type of database (relational, hierarchical, etc.), and any access method (user interface/syntax, such as SOL) can be implemented. The DBMS effectively takes on the form of any desired DBMS, even though the underlying system of atoms and molecules remains unchanged.

The present invention further supports first normal form databases by ensuring that any data value is stored just once (in a content atom) and never duplicated. This feature provides the basis for building first normal form databases.

The present invention also allows for dynamic schema evolution. Any, or all, attributes of a type atom may be changed at will. For example, a type atom for "Humidity" may be changed, thereby changing the description of all instances of "Humidity" without affecting the associated content atoms or any of the instance atoms. This is a significant advantage over conventional DBMS's, where restructuring is required when changing record layouts, adding or removing fields, etc. Molecules (fields) may be added, change format, or erased from an inner relation (record) without affecting any other part of the database. The need for database restructure, unload/load, etc., is eliminated, reducing the cost and complexity of data administration and avoiding downtime.

The present invention also eliminates the need for index fields. Instead, an integrated search structure based on content atoms provides the ability to quickly search for a specific data value. The search structure of the present invention (as described above with reference to Figure 9) is more robust than indexes, which are typically maintained in separate files. Thus the search structure of the present invention reduces the complexity of database design and administration.

The present invention has several performance and storage advantages over prior art systems. An instance atom may be disconnected, and reconnected, without touching the associated type or content atoms. For example, an instance of "John Smith" could be disconnected from type atom "Customer" and reconnected to type atom "Supplier" in one step. In a conventional DBMS, this operation would require the deletion of one record and the creation of another record, a much slower process.

Each data value/property (i.e. "John Smith") will occur only once within the database, in a content atom. This feature saves storage space. In addition, changing all instances with data content "John Smith" to "Tom Smith" requires just one transaction, improving system performance dramatically. Performance is also improved because a search for "John Smith" requires the identification of just one value, which then automatically provides all instances with the data content "John Smith." In addition, empty fields in records (i.e. fields that have a null-value) do not exist in the present invention, thus saving space and improving performance.

Complex data structures can be implemented as a "native" part of the database instead of requiring complex manipulation and transformation to fit into the database structure. This capability improves performance by avoiding transformation processes for each read and write, and also produces storage space savings. Further performance and storage savings are found in the present invention because history information is stored at the instance atom level, rather than at the record or object level.

Although the invention has been described with a certain degree of particularity, it should be recognized that elements thereof may be altered by persons skilled in the art without departing from the spirit and scope of the invention. The invention is limited only by the following claims and their equivalents.

What Is Claimed Is:

1. A memory for storing data for access by a program executing in an information handling system, comprising:

a data structure stored in said memory, said data structure including information resident in a database used by the program and comprising:

a plurality of content elements, wherein each content element contains a unique data item;

a plurality of type elements, wherein each type element stores type data, and wherein each type element includes a unique type handle; and

a plurality of instance elements, wherein each instance element links one content element and one type element.

2. A memory according to claim 1, wherein each data item is uniquely stored in only one content element.

3. A memory according to claim 1, wherein said data structure further comprises one or more inner relations, wherein each inner relation comprises one or more instance elements linked together.

4. A memory according to claim 3, wherein said data structure further comprises one or more outer relations, wherein each outer relation comprises one or more inner relations linked together by a linking means.

5. A memory according to claim 4, wherein the linking means comprises a content atom in a first inner relation containing a pointer to an instance atom in a second inner relation.

6. A memory according to claim 1, wherein said data structure further comprises one or more history links, wherein each history link links a new instance element to an old instance element.

7. A memory according to claim 1, wherein said data structure further comprises a data dictionary, wherein said type elements are stored in said data dictionary.

8. A memory according to claim 1, wherein said data structure further comprises a search structure, said search structure comprising:

one or more mini-structures, wherein each mini-structure contains one or more content elements linked together; and

for each mini-structure, a vector comprising one or more bytes of data, wherein said vector determines which mini-structure to search for a desired content element.

9. A computer-readable medium for storing data for access by a program executing in an information handling system, comprising:

a data structure stored on said computer-readable medium, said data structure including information resident in a database used by the program and comprising:

a plurality of content elements, wherein each content element contains a unique data item;

a plurality of type elements, wherein each type element stores type data, and wherein each type element includes a unique type handle; and

a plurality of instance elements, wherein each instance element links one content element and one type element.

10. A computer-readable medium according to claim 9, wherein each data item is uniquely stored in only one content element.

11. A computer-readable medium according to claim 9, wherein said data structure further comprises one or more inner relations, wherein each inner relation comprises one or more instance elements linked together.

12. A computer-readable medium according to claim 11, wherein said data structure further comprises one or more outer relations, wherein each outer relation comprises one or more inner relations linked together by a linking means.

13. A computer-readable medium according to claim 12, wherein the linking means comprises a content atom in a first inner relation containing a pointer to an instance atom in a second inner relation.

14. A computer-readable medium according to claim 9, wherein said data structure further comprises one or more history links, wherein each history link links a new instance element to an old instance element.

15. A computer-readable medium according to claim 9, wherein said data structure further comprises a data dictionary, wherein said type elements are stored in said data dictionary.

16. A computer-readable medium according to claim 9, wherein said data structure further comprises a search structure, said search structure comprising:

one or more mini-structures, wherein each mini-structure contains one or more content elements linked together; and

for each mini-structure, a vector comprising one or more bytes of data, wherein said vector determines which mini-structure to search for a desired content element.

17. An information handling system, comprising:

one or more processors;

one or more images of an operating system for controlling the operation of said processors;

one or more programs executing in said processors;

memory means; and

a data structure stored in said memory means, said data structure including information resident in a database used by said programs, and comprising:

a plurality of content elements, wherein each content element contains a unique data item;

a plurality of type elements, wherein each type element stores type data, and wherein each type element includes a unique type handle; and

a plurality of instance elements, wherein each instance element links one content element and one type element.

18. An information handling system according to claim 17, wherein each data item is uniquely stored in only one content element.

19. An information handling system according to claim 17, wherein said data structure further comprises one or more inner relations, wherein each inner relation comprises one or more instance elements linked together.

20. An information handling system according to claim 19, wherein said data structure further comprises one or more outer relations, wherein each outer relation comprises one or more inner relations linked together by a linking means.

21. An information handling system according to claim 20, wherein the linking means comprises a content atom in a first inner relation containing a pointer to an instance atom in a second inner relation.

22. An information handling system according to claim 17, wherein said data structure further comprises one or more history links, wherein each history link links a new instance element to an old instance element.

23. An information handling system according to claim 17, wherein said data structure further comprises a data dictionary, wherein said type elements are stored in said data dictionary.

24. An information handling system according to claim 17, wherein said data structure further comprises a search structure, said search structure comprising:

one or more mini-structures, wherein each mini-structure contains one or more content elements linked together; and

for each mini-structure, a vector comprising one or more bytes of data, wherein said vector determines which mini-structure to search for a desired content element.

25. An information handling system according to claim 17, further comprising means for adding a data item to said data structure, said means for adding comprising:

means for receiving the data item;

means for creating a new content element containing the data element;

means for creating a new instance element linked to the new content element; and

means for linking the new content element and an appropriate type element to the new instance element.

26. An information handling system according to claim 25, further comprising:

means for determining if the data item already exists in a content element in the data structure; and

means for linking the new instance element to an existing content element.

27. An information handling system according to claim 17, further comprising means for searching the data structure to find a desired data item, comprising:

means for selecting a mini-structure to search based on one or more bytes in the desired data item; and

means for searching the mini-structure to find the desired content element.

28. An information handling system according to claim 17, further comprising means for updating a content element in said data structure, said means for updating comprising:

means for creating a new content element;

means for creating a new instance element;

means for linking the new content element to the new instance element; and

means for linking the new instance element to an old instance element.

29. An information handling system according to claim 17, further comprising means for updating a content element in said data structure, said means for updating comprising:

means for changing a data item in the content element; and

means for relinking the changed content element to an existing instance element.

30. A method for managing data in a database, comprising the steps of:

creating a plurality of content elements, wherein each content element contains a unique data item;

creating a plurality of type elements, wherein each type element stores type data, and wherein each type element includes a unique type handle; and

creating a plurality of instance elements, wherein each instance element links one content element and one type element.

31. A method according to claim 30, wherein each data item is uniquely stored in only one content element

32. A method according to claim 30, further comprising the step of adding a new data item to the database, said adding step comprising the steps of:

receiving the data item;

creating a new content element containing the data element;

creating a new instance element linked to the new content element; and linking the new content element and an appropriate type element to the new instance element.

33. A method according to claim 32, further comprising:

determining if the data item already exists in a content element in the database; and

linking the new instance element to an existing content element.

34. A method according to claim 30, further comprising the step of searching the database to find a desired data item, wherein said searching comprises the steps of:

selecting a mini-structure to search based on one or more bytes in the desired data item; and

searching the mini-structure to find the desired content element.

35. A method according to claim 30, further comprising the step of updating a content element in said database, said updating step comprising the steps of:

creating a new content element;

creating a new instance element;

linking the new content element to the new instance element; and

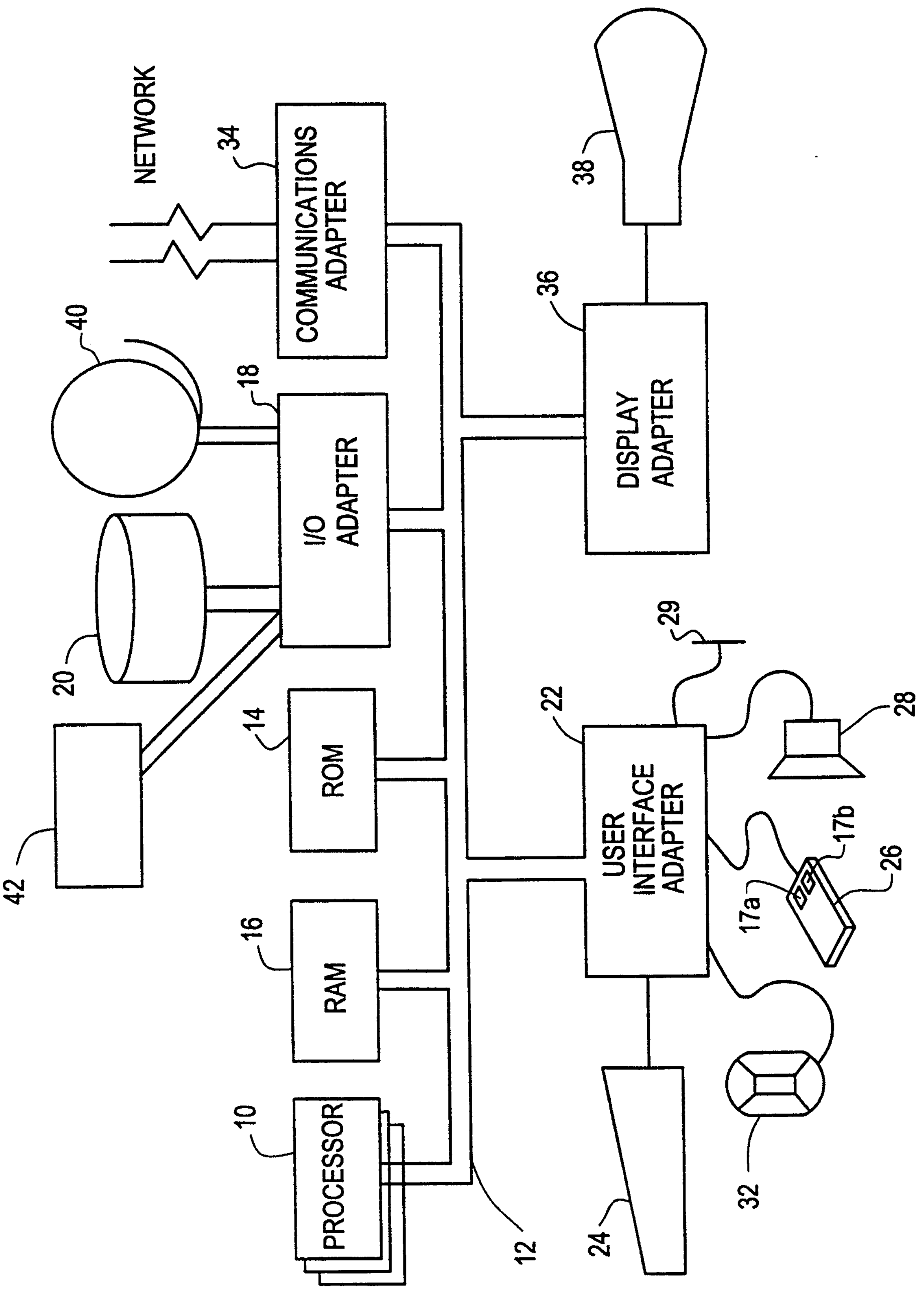
linking the new instance element to an old instance element.

36. A method according to claim 30, further comprising the steps of updating a content element in said database, said updating step comprising the steps of:

changing a data item in the content element; and

relinking the changed content element to an existing instance element.

FIG. 1


NETWORK
34
COMMUNICATIONS ADAPTER
40
18
I/O ADAPTER
20
42
14
ROM
16
RAM
10
PROCESSOR
12
38
36
DISPLAY ADAPTER
29
22
28
USER INTERFACE ADAPTER
17b
26
17a
32
24

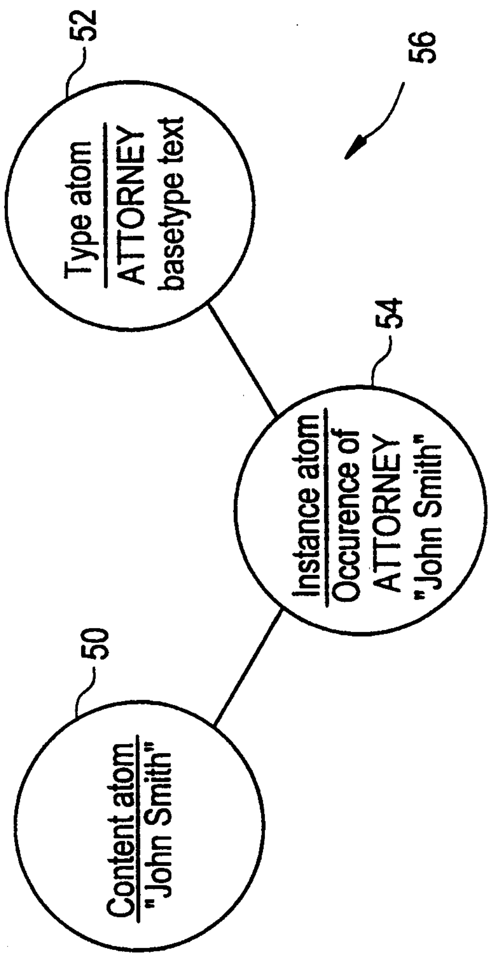

Content atom
"John Smith"
50
Instance atom
Occurence of
ATTORNEY
"John Smith"
54
Type atom
ATTORNEY
basetype text
52
56

FIG. 2

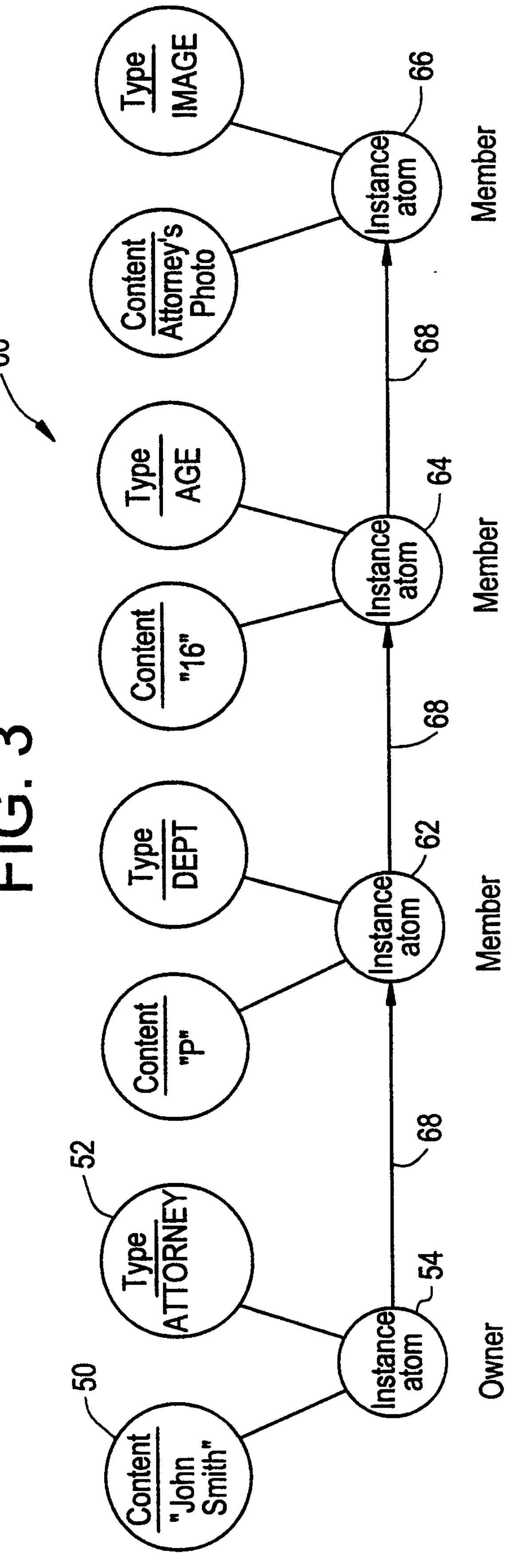

Content
"John
Smith"
50
Type
ATTORNEY
52
Instance
atom
54
Owner
68
Content
"P"
Type
DEPT
Instance
atom
62
Member
68
Content
"16"
Type
AGE
Instance
atom
64
Member
68
Content
Attorney's
Photo
Type
IMAGE
Instance
atom
66
Member
60

FIG. 3

FIG. 4

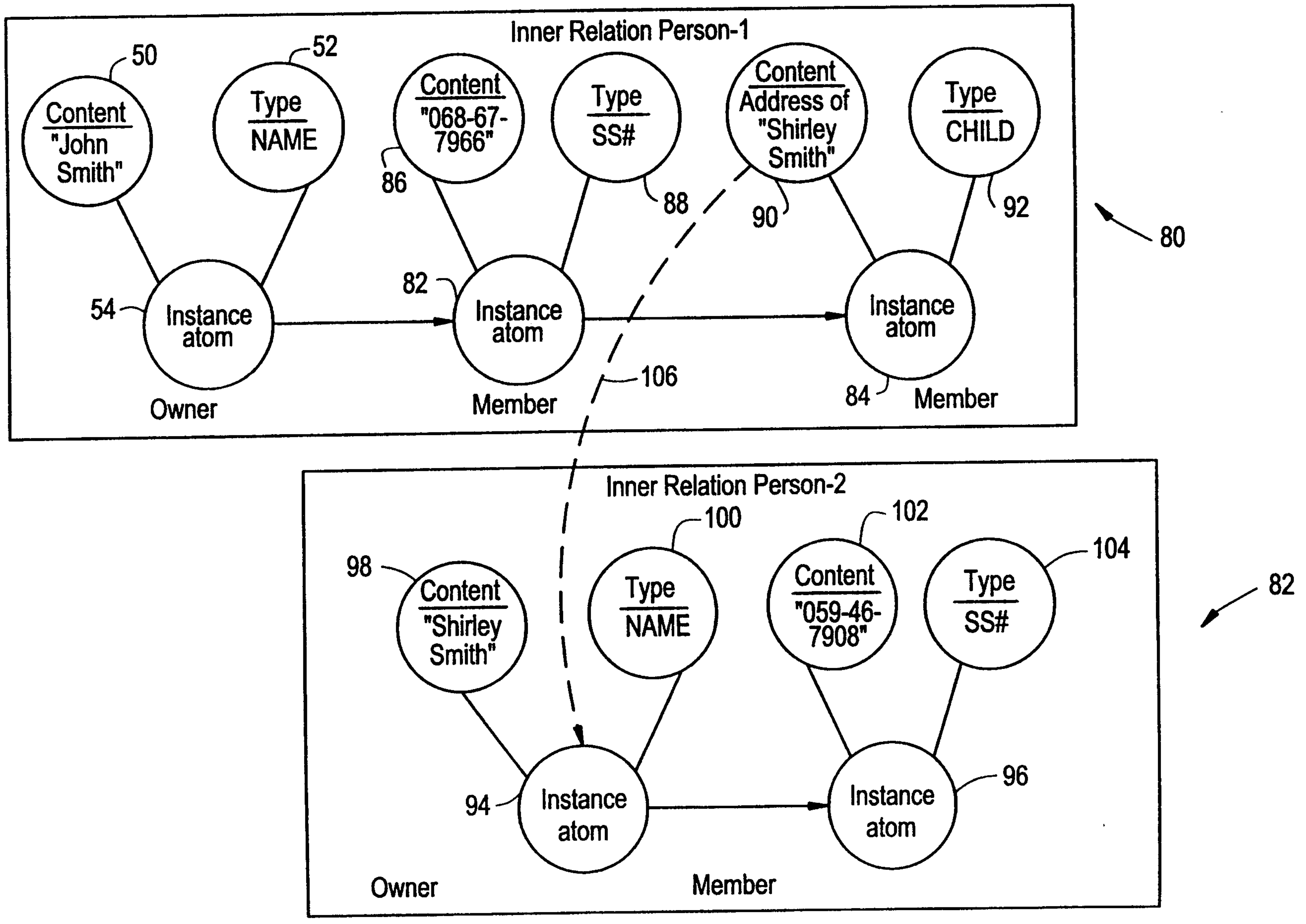

Inner Relation Person-1
Content "John Smith"
50
Type NAME
52
Content "068-67-7966"
86
Type SS#
88
Content Address of "Shirley Smith"
90
Type CHILD
92
54
Instance atom
Owner
82
Instance atom
Member
Instance atom
84
Member
106
80
Inner Relation Person-2
98
Content "Shirley Smith"
Type NAME
100
Content "059-46-7908"
102
Type SS#
104
94
Instance atom
Owner
Instance atom
96
Member
82

FIG. 5

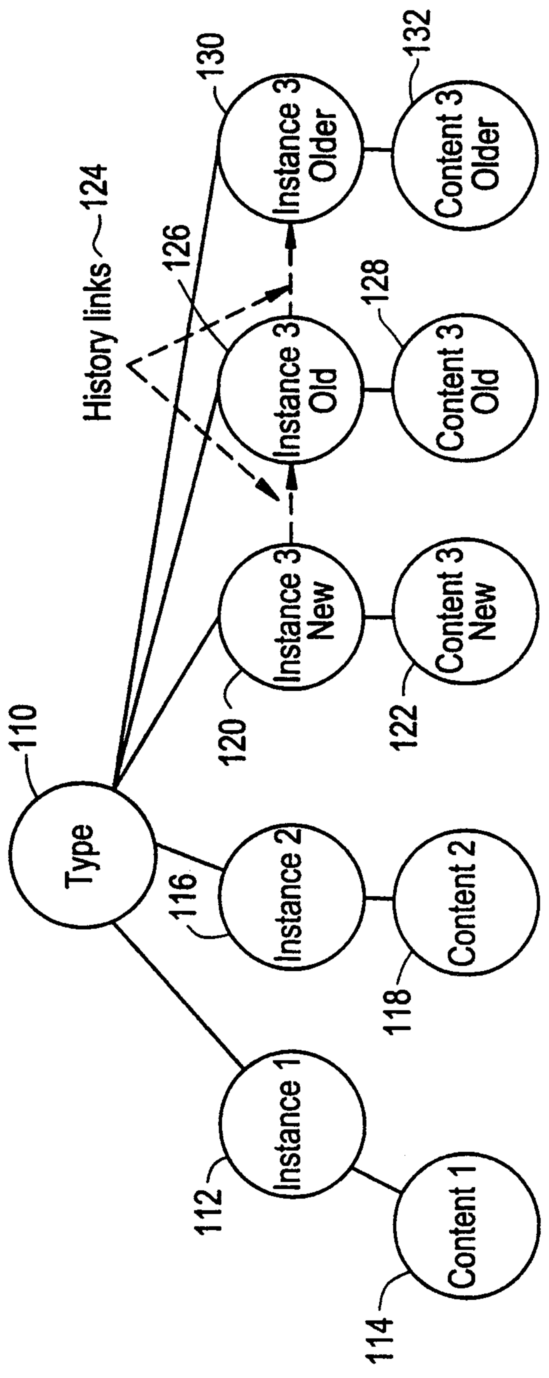

Type
110
Instance 1
112
Content 1
114
Instance 2
116
Content 2
118
Instance 3 New
120
Content 3 New
122
History links
124
Instance 3 Old
126
Content 3 Old
128
Instance 3 Older
130
Content 3 Older
132

FIG. 6

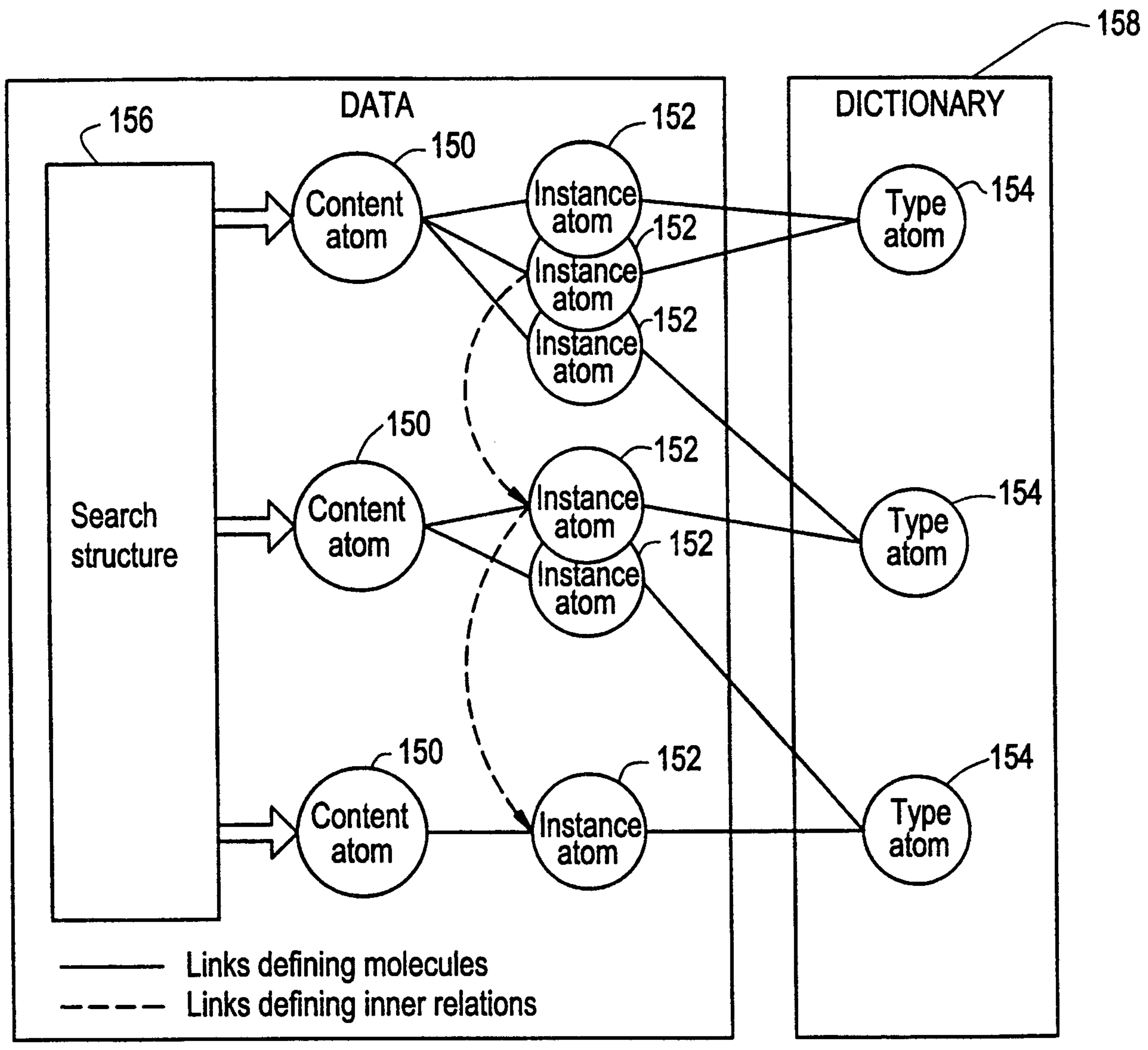

158
DATA
DICTIONARY
156
Search structure
150
Content atom
152
Instance atom
154
Type atom
Links defining molecules
Links defining inner relations

FIG. 7

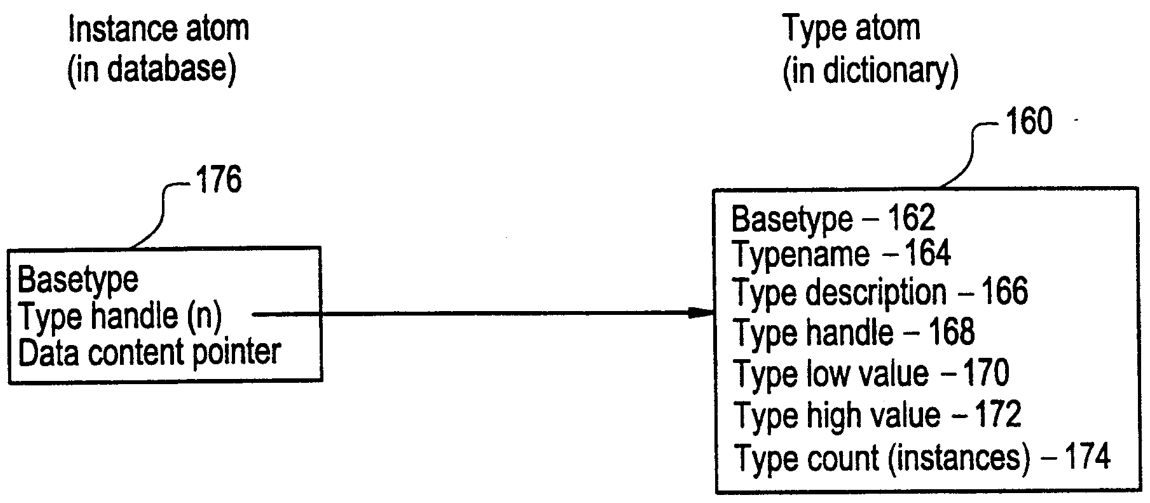

Instance atom
(in database)
Type atom
(in dictionary)
176
Basetype
Type handle (n)
Data content pointer
160
Basetype – 162
Typename – 164
Type description – 166
Type handle – 168
Type low value – 170
Type high value – 172
Type count (instances) – 174

FIG. 8

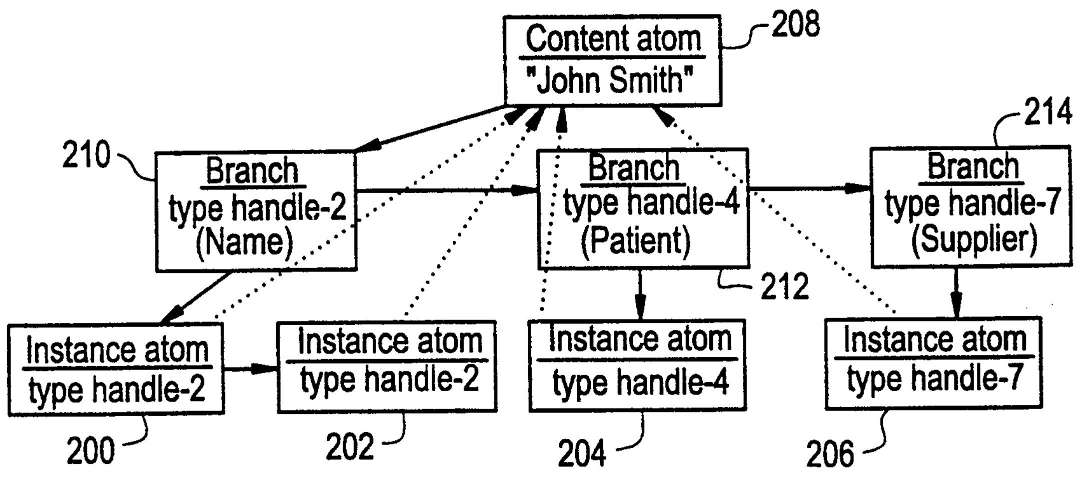

Content atom
"John Smith"
208
210
Branch
type handle-2
(Name)
Branch
type handle-4
(Patient)
212
214
Branch
type handle-7
(Supplier)
Instance atom
type handle-2
200
Instance atom
type handle-2
202
Instance atom
type handle-4
204
Instance atom
type handle-7
206

FIG. 9

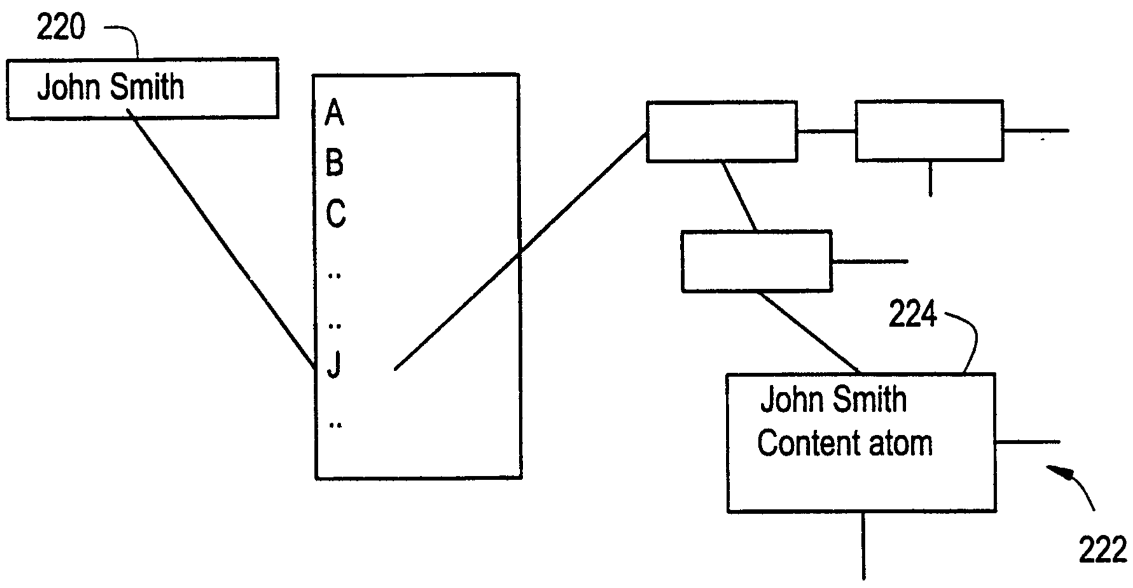

220
John Smith
A
B
C
..
..
J
..
224
John Smith
Content atom
222

FIG. 10

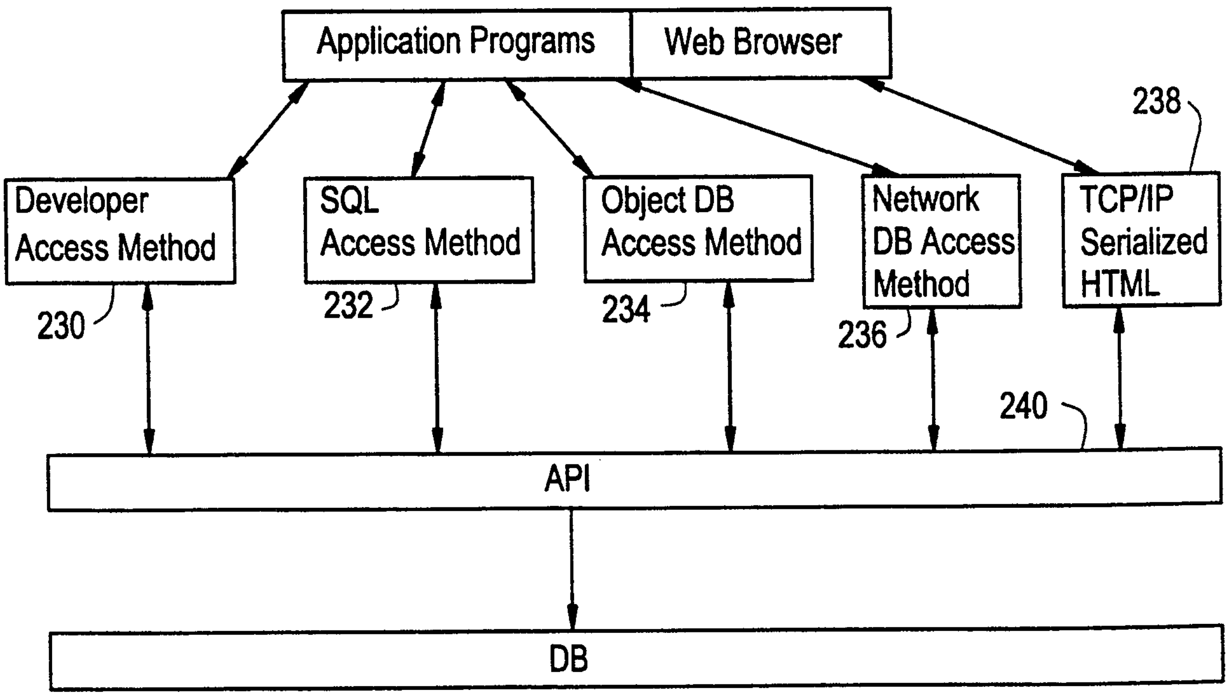
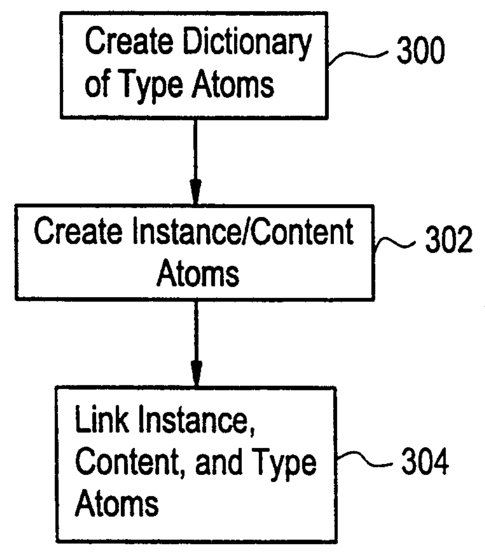
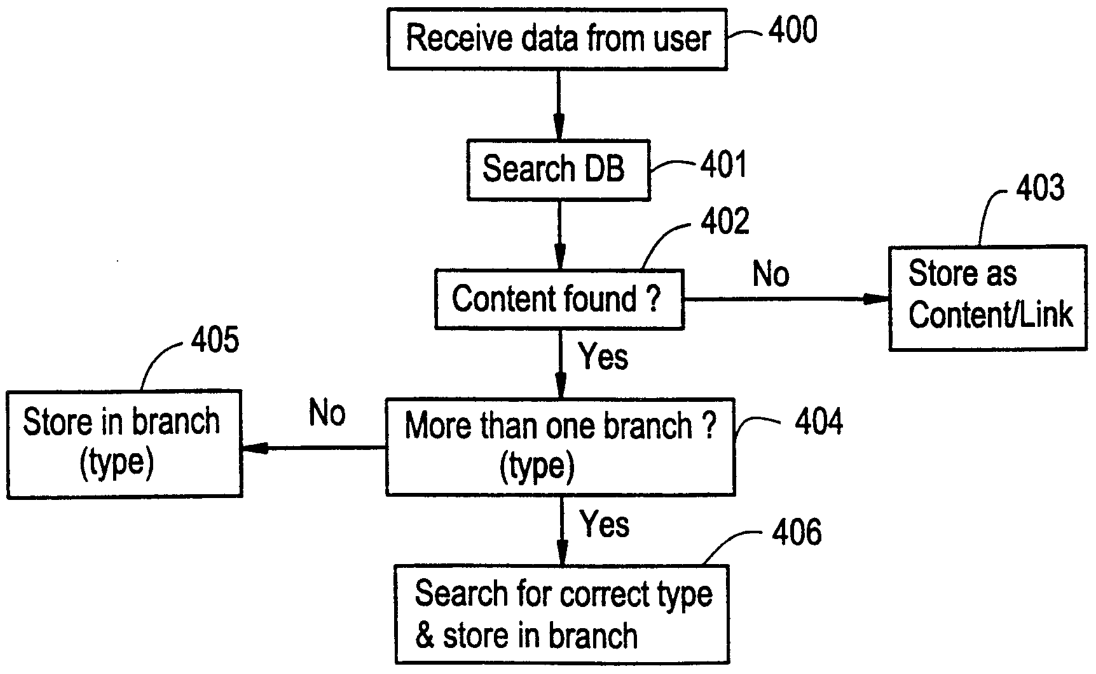

Application Programs
Web Browser
238
Developer Access Method
SQL Access Method
Object DB Access Method
Network DB Access Method
TCP/IP Serialized HTML
230
232
234
236
240
API
DB

FIG. 11



Create Dictionary of Type Atoms
300
Create Instance/Content Atoms
302
Link Instance, Content, and Type Atoms
304

FIG. 12



Receive data from user
400
Search DB
401
402
403
Content found ?
No
Store as Content/Link
Yes
405
Store in branch (type)
No
More than one branch ? (type)
404
Yes
406
Search for correct type & store in branch

FIG. 13

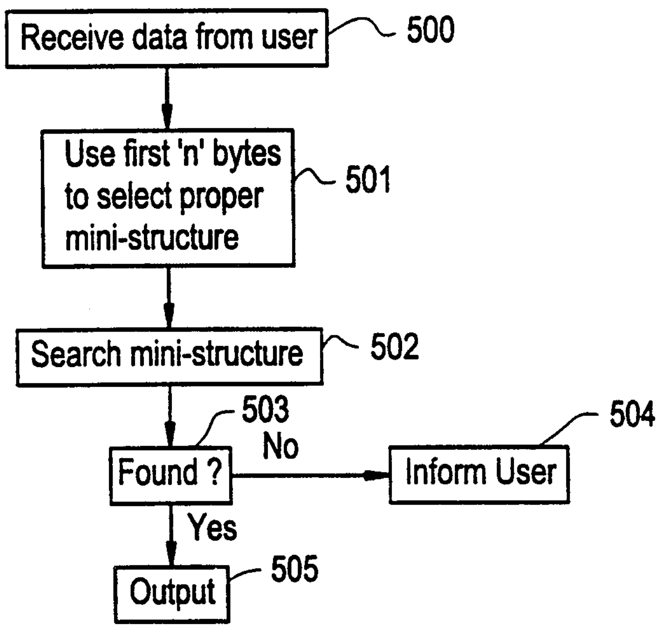
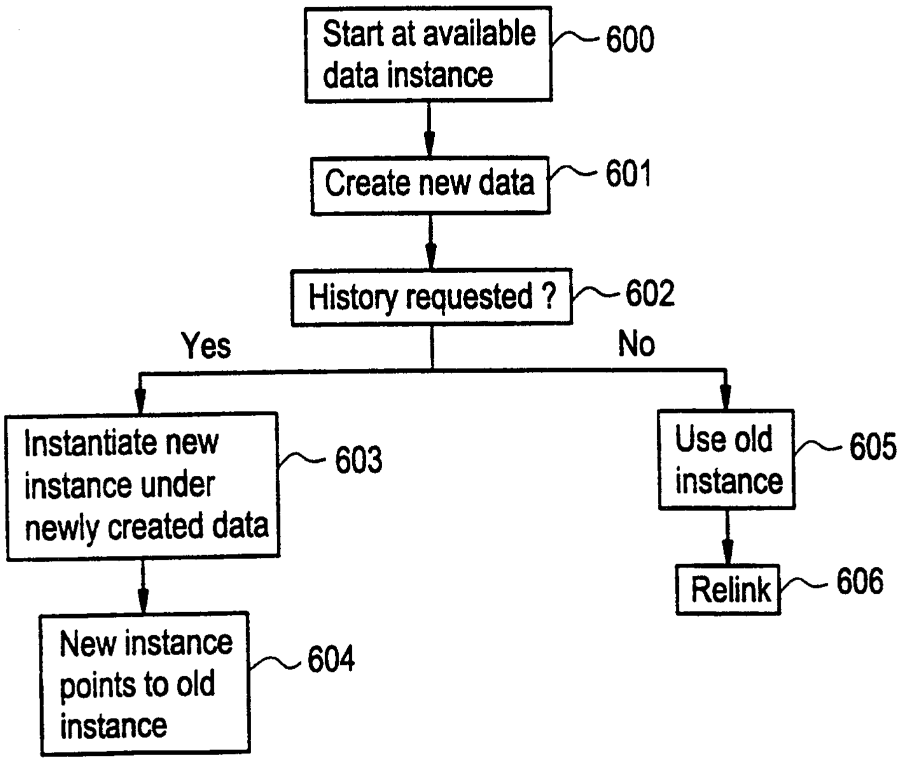

Receive data from user
500
Use first 'n' bytes to select proper mini-structure
501
Search mini-structure
502
503
No
504
Found ?
Inform User
Yes
505
Output

FIG. 14


Start at available data instance
600
Create new data
601
History requested ?
602
Yes
No
Instantiate new instance under newly created data
603
Use old instance
605
Relink
606
New instance points to old instance
604

INTERNATIONAL SEARCH REPORT

International Application No
PCT/NO 98/00139

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 96 18958 A (UFIL UNIFIED DATA TECHNOLOGIES ;AHMADI BABAK (CA)) 20 June 1996 see abstract see page 1, line 1 - page 2, line 14 ---	1,9,17, 30
A	EP 0 229 232 A (TEKTRONIX INC) 22 July 1987 see abstract ---	1,9,17, 30
A	EP 0 216 535 A (TRW INC) 1 April 1987 see abstract -----	1,9,17, 30

☐ Further documents are listed in the continuation of box C.

[✓] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 28 July 1998	Date of mailing of the international search report 05/08/1998
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Katerbau, R

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/NO 98/00139

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9618958 A	20-06-1996	US 5684985 A	04-11-1997
		CA 2206132 A	20-06-1996
		EP 0797806 A	01-10-1997
EP 0229232 A	22-07-1987	JP 1860622 C	27-07-1994
		JP 62160549 A	16-07-1987
		US 5047918 A	10-09-1991
EP 0216535 A	01-04-1987	US 4714995 A	22-12-1987
		JP 3003259 B	18-01-1991
		JP 62111348 A	22-05-1987