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 99/13419

(43) International Publication Date: 18 March 1999 (18.03.99)

(21) International Application Number: PCT/US98/18767

(22) International Filing Date: 9 September 1998 (09.09.98)

(30) Priority Data:
08/926,149 9 September 1997 (09.09.97) US

(71) Applicant: MATRIXONE, INC. [US/US]; Two Executive Drive, Chelmsford, MA 01824 (US).

(72) Inventor: TEWKSBARY, David, E.; 52 Campbello Street, Hudson, NH 03051 (US).

(74) Agent: TOSTI, Robert, J.; Testa, Hurwitz & Thibeault, LLP, High Street Tower, 125 High Street, Boston, MA 02110 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND SYSTEM FOR MANAGING INFORMATION USING A RELATIONAL DATABASE

(57) Abstract

A method and system for managing information by storing object data in a relational database. The method includes the steps of storing a separate data table in a relational database for each of a plurality of data types; associating a unique object identifier (OID) to an object; and defining the object using a plurality of data items. Each of the plurality of data items is one of the plurality of data types. After the object has been assigned an OID, the OID is associated with each of the plurality of data items defining the object. Each of the plurality of data items and the associated OID are stored in respective ones of the data tables according to data type.

METHOD AND SYSTEM FOR MANAGING INFORMATION USING A RELATIONAL DATABASE

Field of the Invention

The invention relates generally to a method and system for information management and more specifically to a method and system for storing object data in a relational database.

Background of the Invention

Known systems for managing information include object databases and relational databases. Object databases store information as objects. An object includes a set of related data elements and a set of routines (termed "methods") for operating on the data elements. For example, an object named "tire" may include data elements such as radius, width and tread. The object named "tire" may also include a method such as a program that calculates the weight of the tire based on information known about the tire (i.e. radius, width, material). Object databases also allow for objects to "inherit" attributes from other objects. For example, if a user wanted to create objects named "bias tire" and "radial tire," the user could create these objects by having them inherit the attributes of the existing object "tire" and then provide additional attributes which make the objects distinct. Inheritance allows a user to create an object that has the qualities of another existing object without having to redefine all of the characteristics of the existing object.

In a relational database, data items are organized according to relations. A relational database stores the data items in tables. The rows of each table represent records or collections of information regarding a particular item and the columns represent fields or attributes of the records. Relationships among the data items stored in the different tables are built by including corresponding fields in more than one table. For example, a data item of a record in one table may be linked to a data item of a record in another table by including an identical field in both data tables and having the information stored in the corresponding field be the same for both records. For example, a relational database for storing tire information may contain a separate table for storing the values of each of the tire characteristics (i.e. separate tables for radius, width, tread, etc.) and pointers defining the relationships between the tables. Relational databases are more well known and accepted in commercial database environments than are object databases.

What is desired then is a system which manages information by storing object data in a relational database. The present invention permits such functionality.

Summary of the Invention

The invention relates to a method for managing information by storing object data in a relational database. The method includes the steps of storing a separate data table in a relational database for each of a plurality of data types; associating a unique object identifier (OID) to an object; and defining the object using a plurality of data items. Each of the plurality of data items is one of the plurality of data types. After the object has been assigned an OID, the OID is associated with each of the plurality of data items defining the object. Each of the plurality of data items and the associated OID are stored in respective ones of the data tables according to data type. In one embodiment, the method includes the steps of storing a method for acting upon the data items in a data table and associating the method with the object. In another embodiment, the method includes the step of creating an additional object by type inheriting the attributes of the existing object type and defining additional attributes for the new object.

The invention also relates to an apparatus for managing information by storing object data in a relational database. The apparatus includes a first memory which stores an object and a unique object identifier (OID) which is associated with the object and a second memory which stores a separate data table in a relational database for each of a plurality of data types. The object is defined by a plurality of data items. Each of the plurality of data items is associated with a respective one of the plurality of data types. Each of the plurality of data items is stored in a respective one of the plurality of data tables in association with the OID.

Brief Description of the Drawings

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1 is a schematic diagram of a computer network system of the type used to practice the invention;

FIG. 2 is a simplified functional block diagram of file servers and workstations on the network system of FIG. 1;

FIGS. 3A and 3B are a block diagram of an embodiment of the data structures of the administrative vault of the present invention;

FIG. 4 is a block diagram of an embodiment of the data structures of the user vault of the present invention;

FIG. 5 is a block diagram of an embodiment of the inheritance of a data structure as practiced with the present invention; and

FIG. 6 is a block diagram of an embodiment of the relationship data structure of the present invention.

Like reference characters in the respective drawn figures indicate corresponding parts.

Detailed Description of the Invention

Referring to FIG. 1, a distributed computer system 10 with which the present invention may function includes one or more workstations 12, 14 and file servers 16, 18 connected by a network 20. An information and document management system according to the invention operates in the environment of the distributed computer system 10. The information and document management system provides a place to store information, ways to move information to and from that place, means for graphically visualizing the information, and control over who may access, use, and modify particular information at a given time.

In general, the file servers 16, 18 provide shared access to data to the workstations 12, 14. Users 22, 24 interface with the workstations 12, 14, and the workstations 12, 14 request and receive data from the file servers 16, 18 which have one or more storage devices (not shown) such as hard disks or tape drives. The workstations 12, 14 and file servers 16, 18 can be computers running any operating system supported by the architecture of the distributed system 10.

Referring to FIG. 2, each file server 16, 18 typically includes an internal bus 26 electrically connecting and allowing communication between a central processor 28, a main memory 30, an input/output controller 32, and a fixed disk 34. In addition to the components of a file server, each workstation typically includes a mouse 36, a keyboard 38, and a display monitor 40 which allow a user to interface with the workstation (i.e., enter information into and receive information from the workstation). In the disclosed embodiment, these components of the workstations and servers operate cooperatively (under the direction of one or more computer programs executed thereby) to store, access, and process data and perform all functions described herein.

The information and document management system according to the invention operates in a homogeneous and/or heterogeneous environment of workstations and file servers connected by a network. In general, any combination of workstations and file servers can be mixed on the network, and although the system with which the present invention may function is described as a distributed system, the invention may also function with a stand alone computer system having data storage capabilities.

The information and document management system according to the invention mainly executes on the workstations; the file servers perform relatively little processing. In the operation of the disclosed information and document management system, the workstations (the "clients") perform most of the processing such as the interfacing (e.g., menu handling, graphical presentation) and database manipulation.

while the file servers (the "servers") store the database information (e.g., meta-data, files) and provide "pages" of the information when requested to do so by the workstations.

The information and document management system according to the invention operates under the assumption that users have "private" space in which they run their applications and store their files. This private space usually is in a user's workstation. It further assumes that users share information (files and the information about them) in "shared" space. Almost all system processes run on the workstations, and the servers provide access to the meta-data, files, and other application-dependent services.

The architecture shown in FIGS. 1 and 2 results in an information and document management system which is relatively insensitive to the number of users on the system. That is, performance of the information and document management system generally does not suffer due to the number of workstations. Performance is primarily a function of the complexity of queries or requests made by the workstations and/or the size of the files or data pages moved by the file servers in response to the queries or requests.

In one embodiment, the information and document management system according to the invention is implemented by one or more computer programs which execute on the workstations and servers described herein. The management system can be implemented in a variety of other ways including with the use of specially-designed electronic hardware.

The information and document management system has a distributed architecture which allows it to be used by individual work groups as well as entire business organizations. The management system is based on a relational database structure. With the management system, information can be widely distributed on the various processors (e.g., workstations and file servers) on the network. Individual databases can be anywhere on the network to suit system management and performance requirements. The workstations and file servers typically maintain exclusive control of files and applications resident therein and thereon, unless files are given over to the management system for management. In general, interactive processes (e.g., menu handling, graphics) of the management system and database manipulation processes of the management system run on each participating workstation while the file servers provide data storage. The management system can coexist and work with applications that operate in the environment of the distributed system 10.

The management system employs a graphical user interface for ease of defining and navigating through a database. A mouse 36 (FIG. 1) is the primary means for a user to interact or interface with the management system. Use of dialogs and keyboard entries are minimized by techniques such as providing on-screen lists of selectable options to users, allowing users to set preferences, and remembering input values from session to session. The management system's graphical interface can include a primary list of menu options with pull-down menus and input dialogs, and a toolbar can provide frequently-used functions as buttons with icons. Users can define their own toolbars to suit their particular activities.

The management system allows construction and exploration of the database via graphical "browser" screen displays that present database objects (defined below) and relationships in a variety of formats. The primary browser screen shows objects (e.g., that have been discovered by initiating a Find operation) in a uniform grid arrangement. Each object is represented by an icon and an object type name, and is labeled with a unique object instance name and a revision identifier. The database also can be explored by selecting an object and thus requesting and receiving in graphical format the objects related to it. In addition to representing objects by icons, the graphical browser screens also can display objects in "thumbnail sketch" form in which a small, raster image of the object is presented.

The management system can be modified or customized by a user to address a variety of business and professional needs. For example, a user can define business or information "objects" as well as "policies" that govern access to and modification of the objects. The management system also provides full control over revisions of objects. It also can provide control over versions of objects.

An object is an element of information, such as a document, which can include, for example, a report, a manual, a schematic, a drawing, or a chart. In the management system, an object is uniquely identified by its type, name, revision, and version identifiers. The object typically contains sub-documents such as pages of text, tables of parts, lists of raw materials required to make the parts, and/or references to a final product which is made by assembling the parts. The sub-documents are computer files generated by one or more application programs.

In one embodiment, the management system can manage objects containing information and files related to and created by computer-aided design (CAD) application programs. Such information and files can define various mechanical components and assemblies in drawing and/or part-list format. The management system can manage objects containing information and files created by one or more application programs such as CAD programs, word processing programs, spreadsheet programs, database programs, and generally any other kind of application program which creates computer files.

The information management system of the present invention requires two types of administration. A first type is for managing the objects stored in the relational database of the information management system. This first type is called "business administration." A second type is for managing the computer systems themselves (i.e., the workstations and servers). This second type is called "system administration." The management system is flexible, it allows users to define objects such that they correspond to the particular business of those users. The administration functions of the management system provide users with a graphical means to create the objects and organization elements that model their particular business. Users do not need any knowledge of computer programming languages to utilize the administration functions provided by the management system. The management system provides a graphical interface which makes administrative functions easy and simple to perform. However, there are some elements of the system that do require programming and system knowledge to set up and modify. These include the tasks of setting up the computers and software, and they must be performed by people with knowledge of computer languages and/or computer science.

Business administration involves defining objects, the policies that govern the behavior of the objects, and the relationships that link the objects together. Business administration is designed to be performed by people (called "business administrators") with an understanding of their business and the practices the users will wish to follow. Business administrators need not have programming or computer science skills.

Business administrators create the object types, the associated policies, and the relationships that together constitute the model of how the users' business operates. With the information management system, the data or information model used in a business is completely modifiable by the business administrator. The management system provides some primitive types of objects from which any number of object types can be defined by the business administrator in the organization's own terms. For each object type, policies are defined in which the behavior of the object is specified. These policies include access rules, states, and approval requirements. Relationships define the different ways objects can be connected. The combination of objects, their associated policies, and permitted relationships is in effect the customer's data and process model for the items under control. To ease start-up, the management system offers optional "profession templates" which provide models of typical objects, policies, and relationships for various types of business activity. These templates can be loaded in a customer's system and can either be used as-is or modified to meet the customer's needs.

The business administrator also is responsible for creating and managing the model of the customer's organization. People, the groups to which they belong, and the roles they play are defined. The people also are issued the appropriate type of access license to the various objects in the management system. Thus, the business administrator can create and define objects and the rules that control them.

FIGS. 3A, 3B and 4 show one embodiment of a relational database structure for storing object data in accordance with the present invention. The relational database structure includes an administrative vault 50 and one or more user vaults 52. As used herein, a "vault" is defined as a set of related data tables. While only one user vault 52 is shown, a system according to the invention typically includes a plurality of user vaults 52. In general, the administrative vault 50 is used to define the different types of objects and the user vaults 52 contain instances of actual objects. Thus the administrative vault 50 is used, for example, to define the object "tire" and the user vault would contain the actual data for various specific tires.

In brief, the administrative vault 50 allows the structure of the relational database (i.e., the various pieces of information to be contained therein and the relationships among these pieces of information) to be defined. This module allows objects to be created and their inter-relationships to be defined. The actual instances of the objects in the database are not, in general, of importance to the administrative vault 50.

The administrative vault 50 is created by a business administrator and contains the definitions of the types of objects which may be created and stored in the relational database. The administrative vault 50 may not be modified by actual users. The user vaults 52 are also created by the business administrator and may be accessed and modified by actual users of the system. The business administrator may create a separate user vault 52 for each physical location in which the system is used. For example, if a business has several offices, a user vault 52 may be created for each office. The business administrator alternatively may also create a separate user vault 52 for each project to be stored in the relational database.

In the tables illustrated in the administrative vault 50, the definition of the data that is stored in an entry is indicated in the left side of each entry and example of what the data may be is indicated in brackets in the right side of each entry.

The administrative vault contains a master Object Identifier (mxOID) table 54, a plurality of data type tables (generally 55), and a plurality of tables which interrelate the data type tables (generally 57). Each object created by the business administrator is assigned a unique object identifier (OID) which is stored in the mxOID table 54 of the administrative vault 50. The mxOID table 54 includes an OID field 56 and a Data Type field 58. Each of the OIDs of the objects created by the business administrator are stored in the OID field 56 of the mxOID table 54. Each OID will only have one entry in the mxOID table 54. For example, the entry OID1 59 will only appear once in the mxOID table 54. The Data Type field 58 indicates what kind of data type the object is. The different types of data which may be stored in the administrative vault 50 include Attribute Types, Business Types, Policy Types, State Types, Relationship Types, and Method Types. Other object types may be defined. The administrative vault 50 contains a separate table for each data type which may appear in the mxOID table 54. For example, in one embodiment, the administrative vault contains an Attribute Type table 60, a Business Type table 62, a Policy Type table 64, and a State Type table 66. Other object tables corresponding to other object types may also be defined. Each of these data type tables will be described in detail below. The OIDs stored in the mxOID table 54 are used to refer objects in the mxOID table 54 to the data type tables 60, 62, 64, and 66 which are used to define the objects. While the references between the tables are indicated with arrows in the figures, the arrows do not indicate real pointers and are only used for illustration purposes.

The Attribute Type table 60 provides a definition for each of the attribute type objects listed in the mxOID table 54. As used herein, an "attribute" is a characteristic or quality which a business object (defined below) may have. The Attribute Type table 60 contains a Name field 67, an OID field 68 which contains the OID of the attribute type object from the mxOID table 54 and an attribute definition field 70. Each of the names stored in the Name field 67 is unique. The OID field 68 is used to link the mxOID table

54 to the Attribute Type table 60. For example, OID1 59 is listed as an attribute type object in the mxOID table 54. OID1 59 also appears in the OID field 68 of the Attribute Type table 60 along with a definition 72 of the attribute type. The attribute type may be defined as an integer, a real number, a Boolean, a string, or a date and time. Other attributes are contemplated. In another embodiment, the Attribute Type table 60 also includes a field for a default value of the attribute. Other fields are also contemplated.

The Business Type table 62 provides a definition for each of the business type objects listed in the mxOID table 56. A business type object defines something that the user's business is involved with. For example, a business type object could be a "tire" if the business were involved with selling tires. Such a business type object is just generally a name of the object of interest and contains no information about a real object. For example, the object "tire" does not contain any information about tires. The Business Type table 62 includes a Name field 73, an OID field 74 which contains one of the OIDs from the mxOID table 54 and a definition field 76. Each of the names stored in the Name field 73 is unique. The OID field 74 is used to link the mxOID table 54 to the Business Type table 62. The definition field holds the definition of the object. Each of the OIDs from the mxOID table 54 identified as a business type data type in the Data Type field 58 appear only once in the Business Type table 62.

An attribute type object describes a characteristic of a business type object. For example, characteristics of a tire include the radius of the tire, the width of the tire, and whether the tire has tread. These characteristics are implemented in the administrative vault 50 by associating the business type object "tire" with attribute type objects "radius," "width," and "tread."

Business types may have associated attribute types, policy types, and method types. The Business Type table 62 is linked to the Attribute Type table 60 through a Derived Information Table 78. The Derived Information Table 78 includes a Business Type OID field 80 for storing the OIDs of the business types listed in the OID field 74 of the Business Type table 62, a definition field 82 which lists the definition of the attribute being associated with the business type object, and an Attribute Type OID field 84 for storing the OID of the attribute type being associated with the business type object. For example, OID2 86, the OID for the object "tire", appears once in the OID field 74 of the Business Type table 62 but several times in the Business Type OID field 80 of the Derived Information Table 78. A separate entry is present in the Business Type OID field 80 of the Derived Information Table 70 for each attribute the business type object includes. For example, in Derived Information Table 78, OID2 86 appears three times, once for each attribute object (radius, OID1; width, OID3; and tread, OID2000) which the business type object needs for its description.

In one embodiment, the administrative vault 50 also contains an mxVault table (not shown). The mxVault table has a structure similar to the structure of the Attribute Type table 60 and the Business Type table 62. The mxVault table stores the OIDs of each user vault 52. One purpose of the mxVault table is to allow users to access objects stored in any of the user vaults and to create relationships between business objects located in different user vaults.

As described above, policies govern access and modification of the business type objects. Policies may include access rules, states, and approval requirements. The Policy Type table 64 provides a definition of each of the policy types listed in the mxOID table 54. The Policy Type table 64 works in conjunction with the State Type table 66 to impose state transitioning requirements on business type objects. In one embodiment, all business type objects reference policy type objects. The Policy Type table 64 includes a Name field 87, an OID field 88 which contains one of the OIDs from the mxOID table 54, and a policy definition field 90. The Policy Type table 90 is linked to the Business Type table 62 and the State Type table 66 through a Policy Information table 92. The Policy Information table 92 includes a Policy Type OID field 94 for storing the OIDs of the policy type objects, a Data Type field 96 which defines the data type table being linked to (i.e. the Business Type table 62 or the State Type table 66), and a Data Type OID field 98 which stores the OID of the related data type object being associated with the policy type object. For example, the first row 100 of the Policy Information table 92 relates the policy type object identified by OID60 to the business type object identified by OID2.

The State Type table 66 provides a definition of the different states that may be associated with a business object. State types are used to define the status of an object within its life-cycle. Examples of state types include planned, working, reviewed, released and obsolete. The State Type table 66 includes a Name field 101, an OID field 102 which contains one of the OIDs form the mxOID table 54, a Policy OID field 104 which refers to the Policy Type table 64, and a descriptor field 106 which provides a definition of the state, i.e. planned, working, reviewed, released or obsolete. The descriptor field may also contain a field for the date and time the state is scheduled to be entered and a field for the date and time the state is actually entered. Business objects move from one state to the next when the conditions for state transition defined in the policy and state type are met. Thus in the present example, the "tires policy" OID60 is related to the business type object "tire" OID2 and has the state types "proposed tire" OID3000 and "released tire" OID3001. Such a "tire policy" would be used, for example, to change a specified tire from a proposed design to a released design for manufacture.

The conditions for state transition include the completion of electronic approvals, the presence of files, the presence of related business objects, and other controls. The conditions are associated with the state types through a State Requirement table 108 and a Signature Requirement table 110. The State Requirement table includes a State Type OID field 112 which stores the OID of the State Type being associated with the state requirement, and a state requirement definition field 114 which stores the definition of the state requirement. Each state type may reference to several state requirements in the State Requirement table 108. The Signature Requirement table 110 includes a State Type OID field 116 which stores the OID of the state type being associated with the signature requirement, and a User field 118 which defines what kind of user may provide the signature to fulfill the signature requirement. In one embodiment, the User field 118 contains an OID which links the Signature Requirement table 118 to a User Type table (not shown). The user type is another data type which may be stored in the administrative vault 50. Thus in the present example a "tire" object may transition from a "proposed tire" to a "released tire" upon the completing of a "testing" state requirement 110 and the completion of the signatures requirement by the engineering manager.

As described above, business types may have associated methods. Methods are computer programs which may be associated with a business type object. An example of a method is a program which displays a picture of a business object on the user's computer screen. Continuing the tire example from above, this program would display a picture of a selected tire instance (i.e. XYZ1 tire) on the user's screen.

In one embodiment, the administrative vault 50 stores methods as a data type and includes a separate Method Type table. The Method Type table has a structure similar to the Attribute Type table 60 and the Business Type table 62 described above. Two types of methods that may be associated with a business type object are internal and external. An internal method is a computer program that is stored in the relational database system. An external method is a computer program that is external to the relational database system, but may be executed by the user's computer. Examples of external methods include Microsoft Word and Word Perfect. For example, if the business object is a document, the user may invoke Microsoft Word to edit the document. The Method Type table stores a reference to the external method.

In one embodiment, to invoke a method, the user instructs the system to run a specific method against a specific business object. In another embodiment, the user is accessing a business object and then specifies a method to be run against the business object. In one such embodiment, the user specifies the name of the method to be executed. The system searches the Method Type table in the administrative vault 50 for the OID associated with the method name and then executes the method. For example, if the user is accessing the business object "XYZ1 tire", the user may execute a method named "circumference" to calculate the circumference of the XYZ1 tire. The system searches the Method Type table for the OID associated with the method "circumference" to locate the program and obtains the data needed for the circumference calculation from the data stored in the user vault 52 for the "XYZ1 tire". The user does not need to specify the value for the radius of the "XYZ1 tire" for the circumference calculation because this value is already stored in the Attribute Instance table 146 for the "XYZ1 tire".

Thus, to further illustrate the relationships between the tables in the administrative vault 50, using the example of a business type object "tire" created by the business administrator is now described. The tire company may want to create a database of different tires. To do this, the business administrator creates a business type object named "Tire." Generally, the business administrator decides what qualities of the tires the database will contain; defines the policies which govern the tires; and defines the different states of development a tire may go through. After the business object is created, a database including particular types of tires may be built using the requirements for a tire defined by the business administrator.

In more detail and referring to the administrative vault 50, a business type object "Tire" is stored in the tables of the administrative vault 50. The business type object "Tire" is assigned a business type OID (OID2 86) in the mxOID table 54. This OID is also entered in the Business Type table 62. The mxOID table 54 also contains the OIDs of several attribute data types which are associated with the business type OID (OID2) to define the characteristics that tires have. For example, a tire may be defined by its radius, width, and whether or not it has a tread. These attributes are each assigned an OID in the mxOID table 54 and an entry is formed the Attribute Type table 60. For example, the attribute "radius" is assigned OID 1, the attribute "width" is assigned OID3 and the attribute "tread" is assigned OID2000. The definition field 70 of the Attribute Type table 60 provides a definition of each of the attribute objects. For example, the attribute "radius" is defined as an integer, the attribute "width" is defined as a real number, and the attribute "tread" is defined as a Boolean value, that is the tire either has tread or does not have tread. The attributes are associated with the business type object "Tire" through the Derived Information Table 78. For example OID2 assigned to the business object type "Tire" appears in the Business Type OID field 80 and OID1 assigned to the attribute type "radius" appears in the Attribute Type OID field 84, thereby associating the attribute "radius" with the business object "Tire." The Derived Information Table 78 also includes entries linking OID2 to the OIDs of the width and tread attributes (OID3 and OID2000 respectively.)

The business type object "Tire" is also associated with a policy and states which define the life cycle of such a tire. In the Policy Type table 90, OID 60 refers to the policy associated with the tire. The tire policy is linked to the business object type "Tire" through the Policy Information table 92. For example, the first row of the Policy Information table 92 contains the policy OID (OID 60) in the Policy Type OID field 94 and the OID of the business object type "Tire" (OID2) in the Data Type OID field 98, thereby associating the tire policy with the business object type "Tire". The different states the tire may transition through are defined by the State Type table 66. The Policy Information table 92 also links the OIDs of the state types (i.e. OID3000 assigned to the state "proposed" and OID3001 assigned to the state "released") to the tire policy (OID 60). Once all the object definitions have been created in the administrative vault, a user vault 52 is then defined to create a database of specific tires based upon these definitions.

FIG. 4 is a diagram illustrating the tables located in the user vault 52. As described above, the user vault 52 contains actual instances of object types defined in the administrative vault 50. Continuing the tire database example above, the user vault 52 contains instances of particular tires. The user vault 52 includes an lxOID table 130, data type tables and instance tables. The lxOID table 130 contains the OIDs of every object stored in the user vault 52. Each object created by a user is assigned a unique object identifier (OID). The lxOID table 130 includes an OID field 132 for storing the OIDs of the objects and a data type field 134 which defines what type of object the OID refers to. The data types stored in the user vault 52 include business types and relationship types. The user vault 52 contains a separate table for each data type which may appear in the lxOID table 130. For example, the user vault 52 contains a Business Object table 136 for the business object data type.

Continuing the example from above, the lxOID table 130 contains OIDs assigned to instances of tires. For example, OID6 is assigned to the XYZ1 brand tire. The Business Object table 136 includes references to other tables which are used to instantiate the instances of the business objects. The Business Object table 136 includes an OID field 138 which stores the OIDs of each business type object listed in the lxOID table 130, an OID Type field 140, a Policy type field 142 and other administrative fields 144. Administrative fields are used to control which users have access to different business objects and to store information regarding the history of the business object. The stored information may include information regarding the creation date of the business object, the user which created the business object, the dates the object was modified. The administrative fields 144 also contain a Name field which stores the name of the business objects, i.e. "XYZ1 tire". The OID Type field 140 links the Business Object table 136 to the Business Type table 62 located in the administrative vault 50. In the example shown, the business object "XYZ1 tire" identified by OID6 is linked to the business type identified by OID2 which provides the definition of a tire and is linked to the attributes a tire has (i.e. radius (OID1), width (OID3), and tread (OID2000)). Each business object has a unique name and a unique OID associated with the business object. The Policy type field 142 links the Business Object table 136 to the Policy Type table 64 located in the administrative vault 50. In the example shown, OID6 assigned to the business object "XYZ1 tire" is linked to the policy identified by OID60 which provides the policy for tires.

A series of "instance" tables are created in the user vault 52 using the references to the administrative vault 50. For example, an Attribute Instance table 146 is generated using the link from the Business Object table 136 in the user vault 52 to the Business Type table 62 in the administrative vault 50. The Attribute Instance table 146 includes a Business Object OID field 148 which stores the OID of the instance of the business object, an Attribute OID field 150 which stores an OID of the attribute in the administrative vault 50 which is included in the Business Type definition of the business object, and a value field 152 which contains the actual parameter value for the specific tire. For example, the first row of the Attribute Instance table 152 contains the OID of the business object "XYZ1 tire" (OID6), the OID of the attribute "radius" (OID1), and a value for the radius (32 inches). The Attribute Instance table 146 contains an entry to store a value corresponding to each of the attributes defined for the object in the administrative vault. For example, the tire attributes also include the width and tread attribute types of the Attribute Type table 60.

In one embodiment, the user vault 52 contains a separate Attribute Instance table 146 for each attribute type. For example, in one such embodiment, the user vault 52 contains a separate Attribute Instance table for attributes which are integers, real numbers, strings, Boolean values, and date and time values. In another embodiment, the user vault 52 only contains an Attribute Instance table 146 for each attribute type which exists in the Attribute Type table 60 in the administrative vault 50. For example, if the Attribute Type table 60 only contains integer and real number type attributes, the user vault 52 will contain two Attribute Instance tables 146, one table for the integers and one table for the real numbers.

A State Instance table 154 is generated using the link from the Business Object table 136 in the user vault 52 to the Policy Type table 66 in the administrative vault 50. The State Instance table 154 includes a Business Object OID field 156 which stores the OID of the business object to which the policy refers, a State Type OID field 158 which stores an OID of a state in the administrative vault 50 which is included in the Policy Type definition, and an information field 160. F or example, the first row of the State Instance table 154 contains the OID of the business object "XYZ1 tire" (OID6), the OID of the state "proposed" (OID3000), and information regarding whether the "XYZ1" tire has satisfied all of the requirements for the "proposed" state. The State Instance table 154 contains similar entries to store information for the "released" state.

Referring to FIG. 5, in one embodiment, the business administrator can derive attributes for a new type object from the attributes of an existing type object. In another embodiment, the business administrator can derive methods for a new type object from the methods associated with an existing type object. Deriving a new type object from an existing type object requires all the attributes associated with the existing type object to be inherited by the new type object. It is possible to add other attributes to the "Inherited Attributes". In this embodiment, the Business Type table 62′ located in the administrative vault 50 includes an additional Parent field 162. The Parent field 162 stores the OID of the business type object from which the attributes are to be inherited. As described above, the Derived Information Table 78′ references the Business Type table 62′ to the Attribute Type table 60′.

For example, the Business Type table 62′ may include a business type object named "round object" (identified by OID5000) which includes the attribute "radius" (identified by OID1) in its definition. The business administrator may wish to create two types of round object which include all of the attributes of the business type object "round object" (OID5000). For example, the business administrator may wish to create a business type object named "tire" (OID5001) and a business type object named "gear" (OID 5002) which include all of the attributes, in this case "radius", of "round object" (OID5000). The inheritance is accomplished by storing the OID of the "round object" (OID5000) in the Parent field 162 of the "tire" entry (OID5001) and "gear" entry (OID5002) in the Business Type table 62′. Thus both objects "tire" and "gear" now have the attribute "radius" inherited from the parent "round object". If a new attribute is later associated with the "parent" object, all of the objects which inherit from the parent object will automatically inherit the additional attribute.

The business administrator may also add attributes specific to the new business type objects "tire" (OID5001) and "gear" (OID5002). For example, the business administrator may want to add the attribute "tread" to the tire business type object (OID5001) and the attribute "number of teeth" to the gear business type object (OID5002). These additional attributes are added in the same manner that an attribute is associated with a business type object in the first instance, an entry in the Derived Information table 78′ is made which links the business type object to the additional attribute. For example, in the second row of the Derived Information table 78′, the business object type "tire" (OID5001) is associated with the attribute type "tread" (OID2000).

In one embodiment, the business administrator may alter the objects stored in the administrative vault 50. For example, the business administrator may add an attribute type object named "price" to the Attribute Type table 60 and associate this attribute with the business type "Tire" (OID2) in the Business Type table 62. Once this association is made, all business objects in the Business Object table 136 in the user vault 52 which are defined by the business object type "Tire" will have the additional attribute "price". For example, the business object "XYZ1 tire" (OID6) would have the additional attribute "price". In one embodiment, the default value for the attribute "price" would be associated with the business object "XYZ1 tire". In another embodiment, the business object "XYZ1 tire" would have a blank entry for the attribute "price" which later may be provided by the user.

In one embodiment, the changes to objects in the administrative vault 50 may be made dynamically without interfering with users that may be accessing the system. The changes are made to the objects such that the next time the user invokes the object, the change will be present. This is possible because the system is interpretive and the characteristics of an object are established each time the object is accessed.

In one embodiment, the business administrator can create relationships between different objects. In this embodiment, the administrative vault 50 includes a Relationship Type table (not shown). The Relationship Type table has a structure similar to the Attribute Type table 60 and the Business Type table 62. The Relationship Type table defines the relationships between different object types. FIG. 6 shows an instance of a Relationship in the user vault 52. The Relationship Object table 164 includes an OID field 166 which contains one of the OIDs from the lxOID table 130, a definition field 168 which defines the type of relationship, a From OID field 170 and a To OID field 172. The OID field 166 is used to correspond the lxOID table 130 to the Relationship Object table 164. The From OID field 170 contains the OID of one of the two business objects being related and the To OID field 72 contains the OID of the second of the two business objects being related. For example, the user may desire to create a relationship between a business object named "XYZ1 Tire" (OID6) and a business object named "XYZ2 tire" (OID100) to define the relationship that both of these tires can be used on the same types of cars. A relationship object could also be used to create a relationship between a business object named "XYZ1" tire and a business object named "ABC1 car" to define the relationship that the XYZ1 tire is a component of the ABC1 car. A single business object may be related to more than one business object by creating several relationship objects which include that business object in the From OID field 170 or the To OID field 72. In another embodiment, relationship objects are used to create a hierarchy of objects.

In one embodiment, a relationship object may relate a business object in a first user vault 52 with a business object in a second user vault 52. In this embodiment, the From OID field 170 is comprised of two fields, a field for the OID of the business object being related and a field for the OID of the vault in which the business object is located. Similarly, the To OID field 172 is comprised of two fields, a field for the OID of the business object being related and a field for the OID of the vault in which the business object is located. For example, if the business object "XYZ1 tire" is located in a different vault than the business type object "XYZ2 tire," the From OID field 170 would contain a field for the OID of the vault of the business object "XYZ1 tire" and a field for the OID of the business object "XYZ1 tire", i.e. OID6. Similarly, the To OID field 172 would contain a field for the OID of the vault of the business object "XYZ2 tire" and a field for the OID of the business object "XYZ2 tire," i.e. OID100.

In one embodiment, to create a new business object, the user specifies the name of the business object type that the user wishes to create and the name of the policy that will govern the business object. For example, to create a new business object for the "XYZ2" brand tire, the user enters the business object type name "Tire" and the policy type name "Tire Policy." The system searches the Business Type table 62 in the administrative vault 50 for the OID associated with the business object type "Tire" (OID2). Once the system has located the OID for the business object type "Tire", the system uses the OID to determine which attributes from the Attribute Type table 60 the new business object "XYZ2 tire" will have. The process for determining which attributes are associated with a business object type was described above in the discussion of FIGS. 3A and 3B. Similarly, the system searches the Policy Type table 64 in the administrative vault 50 for the OID associated with the policy type name "Tire Policy" and uses this OID to determine the policy and state requirements for the new business object "XYZ2 tire."

The system assigns a unique OID to the new business object "XYZ2 tire" and creates a new entry in the lxOID table 130 and the Business Object table 136 for this OID. The system also creates entries in the Attribute Instance table 146 and the State Instance table 154 for the new business object "XYZ2 tire" using the information retrieved from the tables in the administrative vault 50. In one embodiment, the user is prompted to enter values for the different attributes associated with the business object. For example, in creating the business object "XYZ2 tire" the user is prompted to enter values for radius, width and tread, as these are the attributes associated with the business object type "Tire" (OID2). In another embodiment, if the user does not enter a value for an attribute, a default value is entered by the system.

In another embodiment, to retrieve information regarding a specific business object, i.e., "XYZ1 tire", the user enters the name of the business object. As described above, each business object has a unique name. The system searches the Business Object table 136 in the user vault 52 for the OID associated with the business object. For example, the system would search the Business Object table 136 in the user vault 52 for the OID associated with the business object name "XYZ1 tire" and find OID6. Once the system has found the OID associated with the business object, the system may retrieve all the information regarding the business object that is stored in the database (i.e., the radius, width and tread of the XYZ1 tire).

Having described preferred embodiments of the invention, it will now become apparent to one of skill in the art that other embodiments incorporating the concepts may be used. It is felt, therefore, that these embodiments should not be limited to disclosed embodiments but rather should be limited only by the spirit and scope of the following claims.

CLAIMS

What is claimed is:

1. A method for managing data, comprising the steps of:

storing a separate data table in a relational database for each of a plurality of data types;

associating a first unique object identifier to a first object;

defining the first object using a plurality of data items, each of the plurality of data items being one of the plurality of data types;

associating the first unique object identifier with each of the plurality of data items; and

storing each of the plurality of data items and the associated object identifier in respective ones of the data tables according to data type.

2. The method of claim 1, further comprising the steps of:

defining the first object using an additional data item, the additional data item being one of the plurality of data types;

associating the first unique object identifier with the additional data item; and

storing the additional data item and the associated object identifier in one of the data tables according to data type.

3. The method of claim 1, further comprising the step of:

generating, subsequent to the storing of the plurality of data items, a representation of the first object by collecting each data item having the first unique object identifier.

4. The method of claim 1, further comprising the steps of:

associating a second unique object identifier to a second object;

defining the second object using a plurality of data items, each of the plurality of data items being one of the plurality of data types;

associating the second unique object identifier with each of the plurality of data items; and

storing each of the plurality of data items and the associated object identifier in respective ones of the data tables according to data type.

5. A method for managing data, comprising the steps of:

storing a separate data table in a relational database for each of a plurality of data types;

associating a unique object identifier to each of a plurality of objects;

defining each of the plurality of objects using a plurality of data items, each of the plurality of data items being one of the plurality of data types;

for each of the plurality of objects, associating the unique object identifier associated with the object with each of the plurality of data items defining the object; and

storing each of the plurality of data items and associated unique object identifiers in respective ones of the data tables according to data type.

6. A method of building a hierarchy of objects in a relational database, comprising the steps of:

storing a separate data table in a relational database for each of a plurality of data types;

associating a unique object identifier to each of a plurality of objects;

defining each of the plurality of objects using a plurality of data items, each of the plurality of data items being one of the plurality of data types;

for each of the plurality of objects, associating the unique object identifier associated with the object with each of the plurality of data items defining the object;

storing each of the plurality of data items and associated unique object identifiers in respective ones of the data tables according to data type; and

defining a relationship between selected ones of the plurality of objects to create a hierarchy of objects.

7. An apparatus for managing data, comprising:

a first memory storing a first object and an associated object identifier; and

a second memory storing a separate data table in a relational database for each of a plurality of data types;

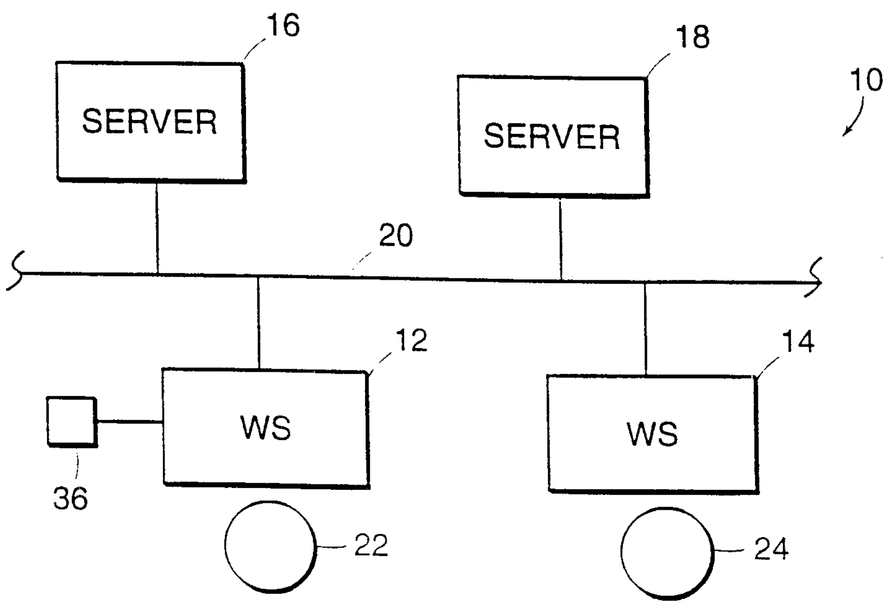
wherein the first object is defined by a plurality of data items, each of the plurality of data items being associated with a respective one of the plurality of data types; and

wherein each of the plurality of data items is stored in a respective one of the plurality of data tables in association with the object identifier.

8. The apparatus of claim 7, further comprising:

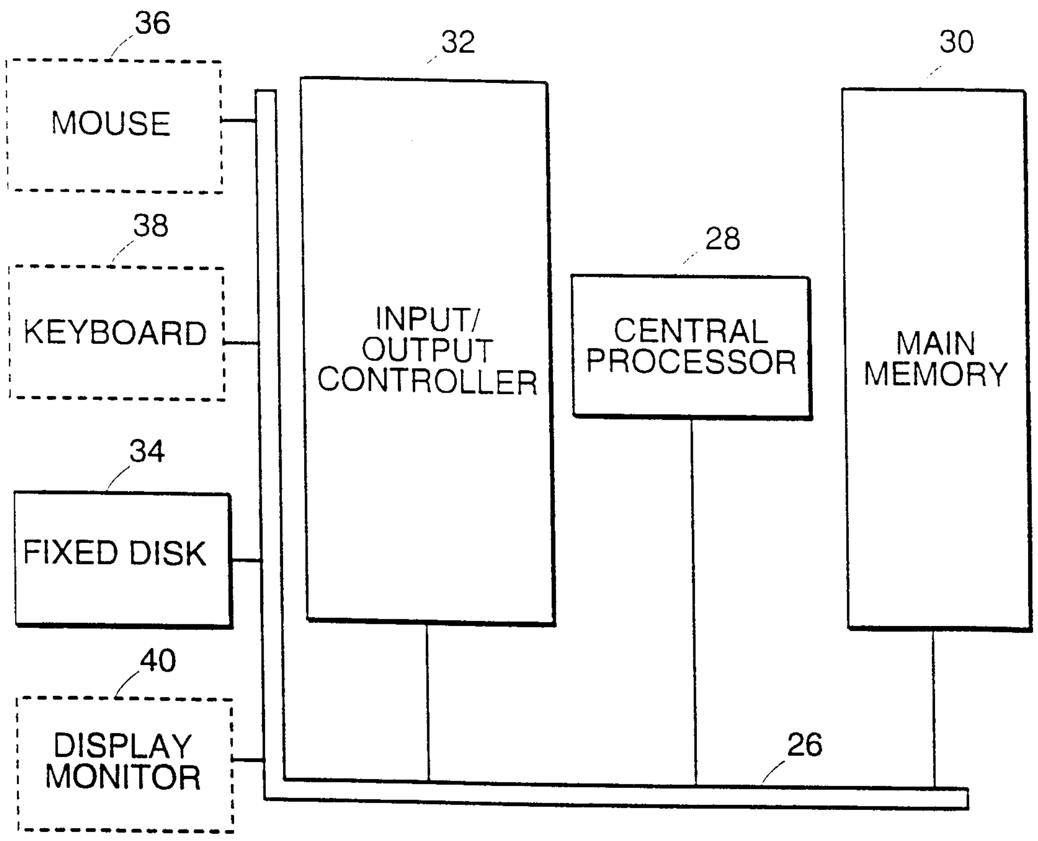
a processor in electrical communication with the first memory and the second memory, the processor generating a representation of the first object by accessing each data item associated with the object identifier.

9. The apparatus of claim 7 wherein the first memory further stores a second object and an indicia of relationship with the first object.



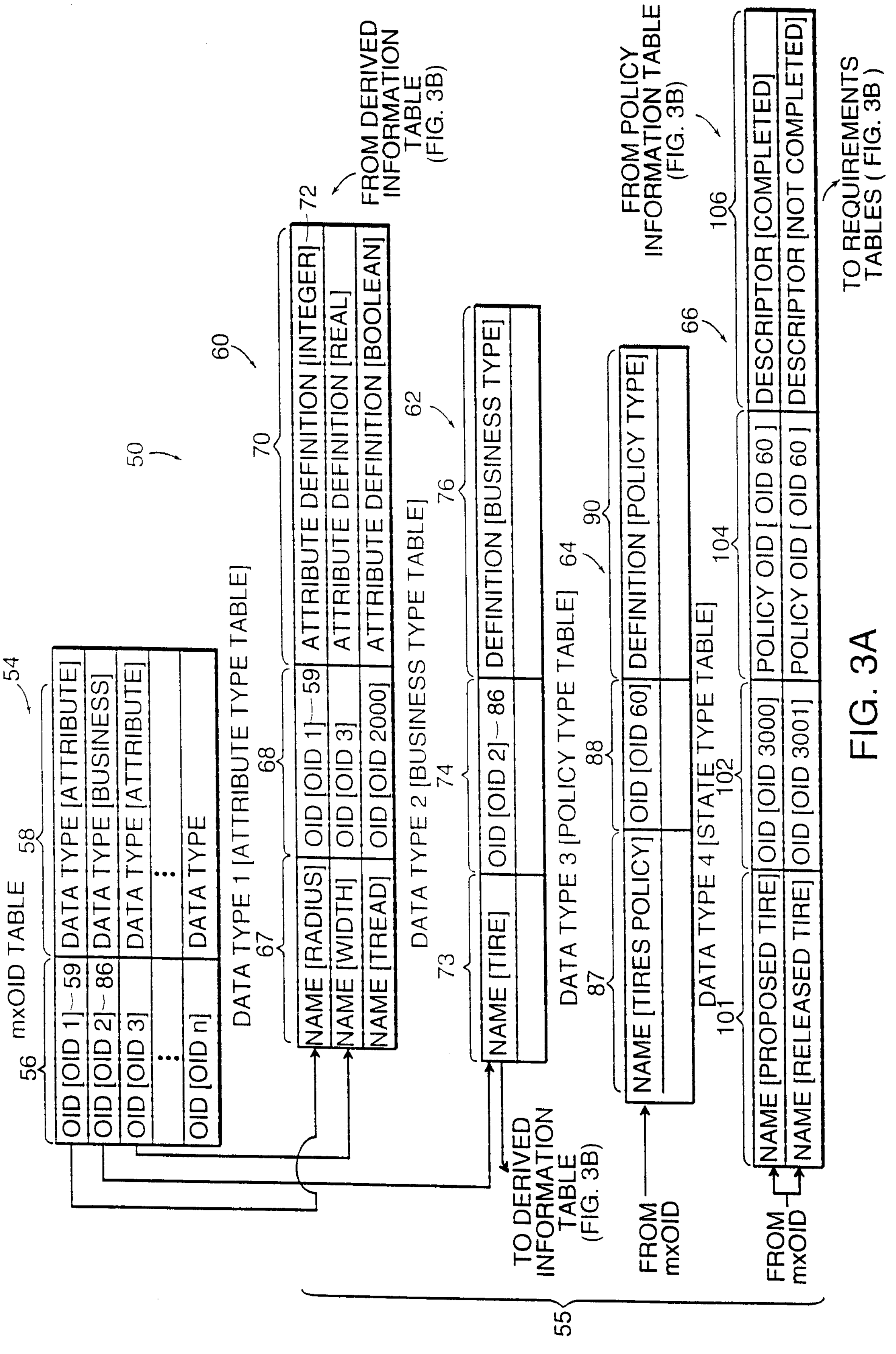
16
18
10
SERVER
SERVER
20
12
14
WS
WS
36
22
24

FIG. 1



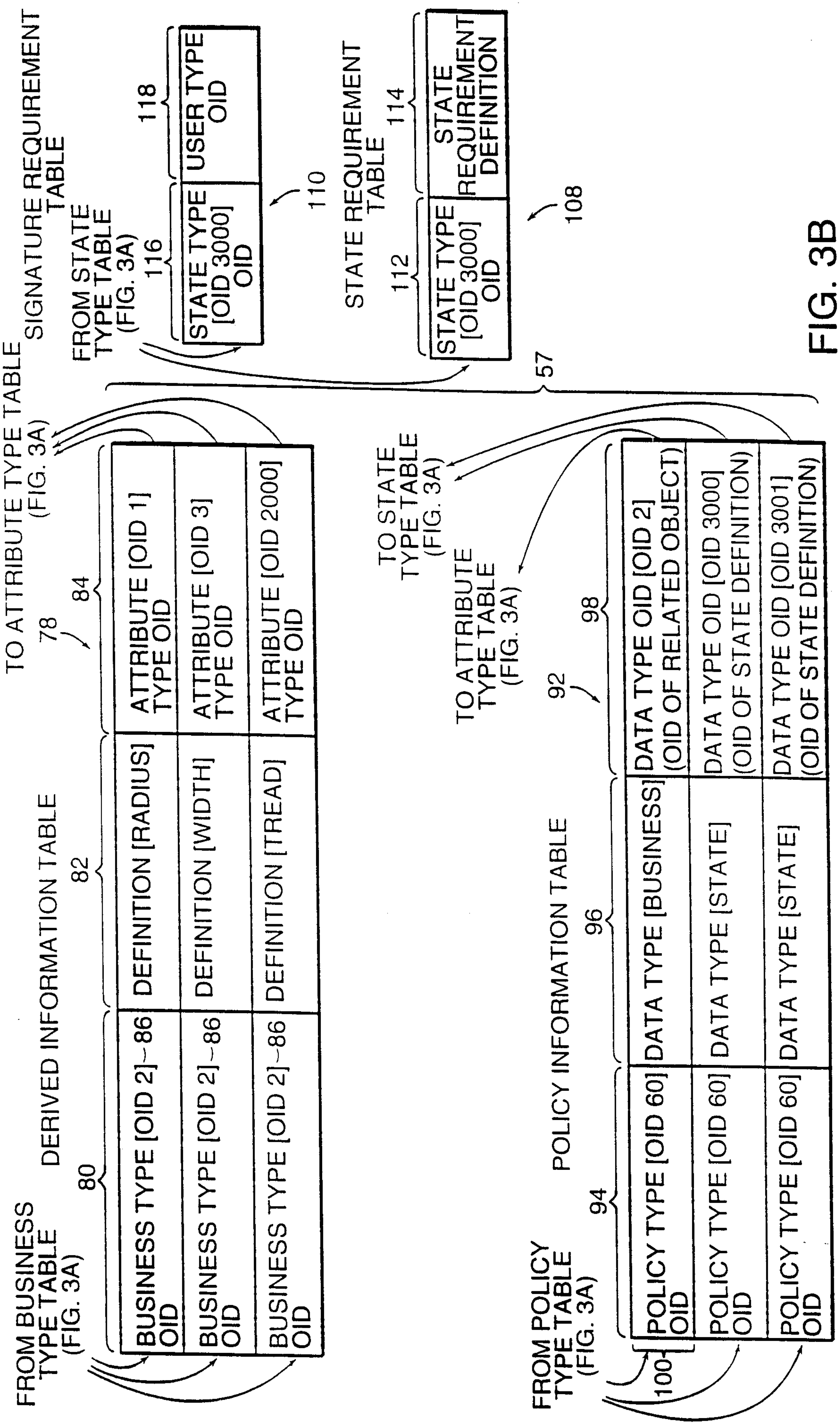
36
32
30
MOUSE
38
28
KEYBOARD
INPUT/ OUTPUT CONTROLLER
CENTRAL PROCESSOR
MAIN MEMORY
34
FIXED DISK
40
DISPLAY MONITOR
26

FIG. 2



50
mxOID TABLE
54
56
58
OID [OID 1]—59 | DATA TYPE [ATTRIBUTE]
OID [OID 2]—86 | DATA TYPE [BUSINESS]
OID [OID 3] | DATA TYPE [ATTRIBUTE]
OID [OID n] | DATA TYPE
DATA TYPE 1 [ATTRIBUTE TYPE TABLE]
60
67
68
70
72
NAME [RADIUS] | OID [OID 1]—59 | ATTRIBUTE DEFINITION [INTEGER]
NAME [WIDTH] | OID [OID 3] | ATTRIBUTE DEFINITION [REAL]
NAME [TREAD] | OID [OID 2000] | ATTRIBUTE DEFINITION [BOOLEAN]
FROM DERIVED INFORMATION TABLE (FIG. 3B)
DATA TYPE 2 [BUSINESS TYPE TABLE]
62
73
74
76
NAME [TIRE] | OID [OID 2]—86 | DEFINITION [BUSINESS TYPE]
TO DERIVED INFORMATION TABLE (FIG. 3B)
DATA TYPE 3 [POLICY TYPE TABLE]
64
87
88
90
NAME [TIRES POLICY] | OID [OID 60] | DEFINITION [POLICY TYPE]
FROM mxOID
DATA TYPE 4 [STATE TYPE TABLE]
66
101
102
104
106
NAME [PROPOSED TIRE] | OID [OID 3000] | POLICY OID [OID 60] | DESCRIPTOR [COMPLETED]
NAME [RELEASED TIRE] | OID [OID 3001] | POLICY OID [OID 60] | DESCRIPTOR [NOT COMPLETED]
FROM POLICY INFORMATION TABLE (FIG. 3B)
TO REQUIREMENTS TABLES (FIG. 3B)
FROM mxOID
55

FIG. 3A



FROM BUSINESS TYPE TABLE (FIG. 3A)
80
DERIVED INFORMATION TABLE
82
TO ATTRIBUTE TYPE TABLE (FIG. 3A)
78
84
BUSINESS TYPE [OID 2]~86
DEFINITION [RADIUS]
ATTRIBUTE [OID 1] TYPE OID
BUSINESS TYPE [OID 2]~86
DEFINITION [WIDTH]
ATTRIBUTE [OID 3] TYPE OID
BUSINESS TYPE [OID 2]~86
DEFINITION [TREAD]
ATTRIBUTE [OID 2000] TYPE OID
BUSINESS OID
SIGNATURE REQUIREMENT TABLE
FROM STATE TYPE TABLE (FIG. 3A)
116
118
STATE TYPE [OID 3000] OID
USER TYPE OID
110
STATE REQUIREMENT TABLE
112
114
STATE TYPE [OID 3000] OID
STATE REQUIREMENT DEFINITION
108
57
TO STATE TYPE TABLE (FIG. 3A)
TO ATTRIBUTE TYPE TABLE (FIG. 3A)
FROM POLICY TYPE TABLE (FIG. 3A)
94
POLICY INFORMATION TABLE
96
92
98
100
POLICY TYPE [OID 60] OID
DATA TYPE [BUSINESS]
DATA TYPE OID [OID 2] (OID OF RELATED OBJECT)
POLICY TYPE [OID 60] OID
DATA TYPE [STATE]
DATA TYPE OID [OID 3000] (OID OF STATE DEFINITION)
POLICY TYPE [OID 60] OID
DATA TYPE [STATE]
DATA TYPE OID [OID 3001] (OID OF STATE DEFINITION)

FIG. 3B

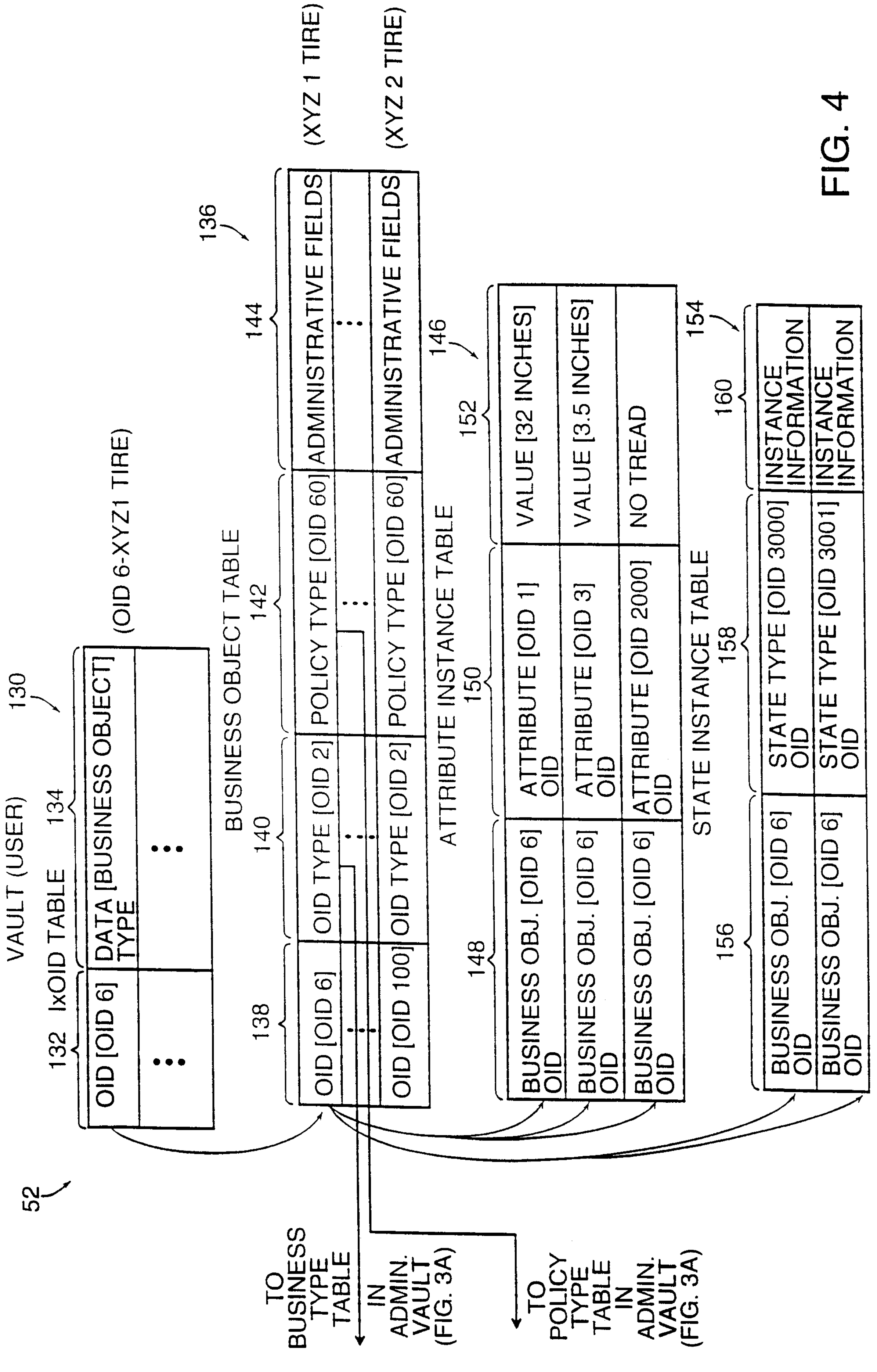

52
VAULT (USER)
130
132 IxOID TABLE 134
OID [OID 6]
DATA [BUSINESS OBJECT TYPE
(OID 6-XYZ1 TIRE)
136
BUSINESS OBJECT TABLE
138
140
142
144
OID [OID 6]
OID TYPE [OID 2]
POLICY TYPE [OID 60]
ADMINISTRATIVE FIELDS
(XYZ 1 TIRE)
OID [OID 100]
OID TYPE [OID 2]
POLICY TYPE [OID 60]
ADMINISTRATIVE FIELDS
(XYZ 2 TIRE)
ATTRIBUTE INSTANCE TABLE
146
148
150
152
BUSINESS OBJ. [OID 6] BUSINESS OID
ATTRIBUTE [OID 1] ATTRIBUTE OID
VALUE [32 INCHES]
BUSINESS OBJ. [OID 6] BUSINESS OID
ATTRIBUTE [OID 3] ATTRIBUTE OID
VALUE [3.5 INCHES]
BUSINESS OBJ. [OID 6] BUSINESS OID
ATTRIBUTE [OID 2000] ATTRIBUTE OID
NO TREAD
STATE INSTANCE TABLE
154
156
158
160
BUSINESS OBJ. [OID 6] BUSINESS OID
STATE TYPE [OID 3000] STATE TYPE OID
INSTANCE INFORMATION
BUSINESS OBJ. [OID 6] BUSINESS OID
STATE TYPE [OID 3001] STATE TYPE OID
INSTANCE INFORMATION
TO BUSINESS TYPE TABLE IN ADMIN. VAULT (FIG. 3A)
TO POLICY TYPE TABLE IN ADMIN. VAULT (FIG. 3A)

FIG. 4

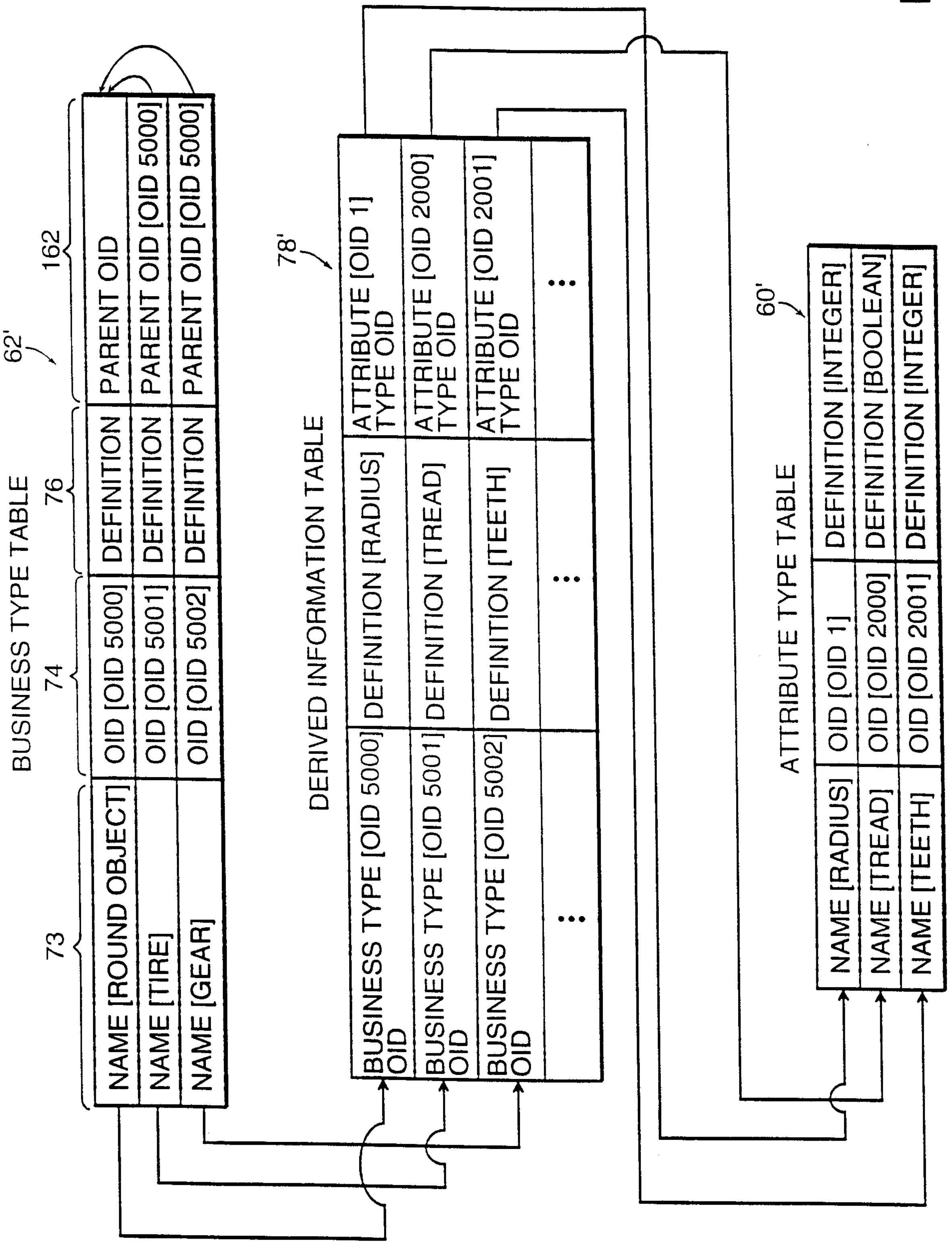

BUSINESS TYPE TABLE
62'
73
74
76
162
NAME [ROUND OBJECT]
OID [OID 5000]
DEFINITION
PARENT OID
NAME [TIRE]
OID [OID 5001]
DEFINITION
PARENT OID [OID 5000]
NAME [GEAR]
OID [OID 5002]
DEFINITION
PARENT OID [OID 5000]
DERIVED INFORMATION TABLE
78'
BUSINESS TYPE [OID 5000] OID
DEFINITION [RADIUS]
ATTRIBUTE [OID 1] TYPE OID
BUSINESS TYPE [OID 5001] OID
DEFINITION [TREAD]
ATTRIBUTE [OID 2000] TYPE OID
BUSINESS TYPE [OID 5002] OID
DEFINITION [TEETH]
ATTRIBUTE [OID 2001] TYPE OID
...
...
...
ATTRIBUTE TYPE TABLE
60'
NAME [RADIUS]
OID [OID 1]
DEFINITION [INTEGER]
NAME [TREAD]
OID [OID 2000]
DEFINITION [BOOLEAN]
NAME [TEETH]
OID [OID 2001]
DEFINITION [INTEGER]

FIG. 5

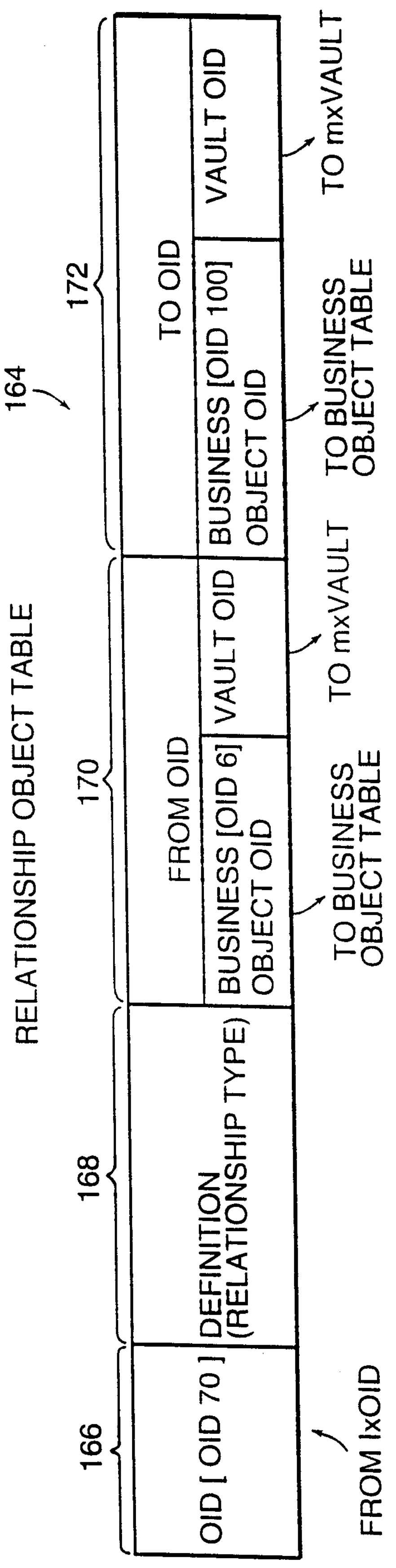

RELATIONSHIP OBJECT TABLE
164
166
168
170
172
OID [OID 70]
DEFINITION (RELATIONSHIP TYPE)
FROM OID
BUSINESS [OID 6] OBJECT OID
VAULT OID
TO OID
BUSINESS [OID 100] OBJECT OID
VAULT OID
FROM IxOID
TO BUSINESS OBJECT TABLE
TO mxVAULT
TO BUSINESS OBJECT TABLE
TO mxVAULT

FIG. 6

INTERNATIONAL SEARCH REPORT

Int ional Application No
PCT/US 98/18767

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	BEECH D ET AL: "OBJECT DATABASES AS GENERALIZATIONS OF RELATIONAL DATABASES" COMPUTER STANDARDS AND INTERFACES, vol. 13, no. 1 / 03, 1 January 1991, pages 221-230, XP000288495 see page 224, left-hand column, paragraph 3 - page 226, left-hand column, paragraph 5 ---	1-9
X	US 5 295 256 A (BAPAT SUBODH) 15 March 1994 see abstract see column 10, line 36 - column 12, line 38; claims; figures 6,14-17,27 ---	1-9
	-/--	

[X] Further documents are listed in the continuation of box C.

[X] 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 14 January 1999	Date of mailing of the international search report 21/01/1999
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 Fournier, C

1

INTERNATIONAL SEARCH REPORT

Int ional Application No
PCT/US 98/18767

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 529 844 A (SUN MICROSYSTEMS INC) 3 March 1993 see column 2, line 45 - column 4, line 11; figures 3-7A	1-9
A	REINWALD B ET AL: "STORING AND USING OBJECTS IN A RELATIONAL DATABASE" IBM SYSTEMS JOURNAL, vol. 35, no. 2, 1 March 1996, pages 172-190, XP000594128 see page 174, left-hand column, line 12 - right-hand column, line 9	1,3-9

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

In ional Application No
PCT/US 98/18767

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5295256 A	15-03-1994	NONE	
EP 0529844 A	03-03-1993	US 5261098 A	09-11-1993
		JP 6324877 A	25-11-1994