Search System and Method based on Multiple, Reusable Ontologies

Kenneth Baclawski, Waltham, MA

Abstract

A distributed computer database system including one or more front end computers and one or more computer nodes interconnected by a network into a search engine for retrieval of objects processed by a variety of interrelated ontologies. Each object conforms to a specific ontology. A query is an object which conforms to a specific ontology, which is to be used for retrieval of objects conforming to one or more target ontologies. A query from a user is transmitted to one of the front end computers which forwards the query to one of the computer nodes, termed the home node, of the search engine. The home node extracts features from the query, according to its ontology. These features are then hashed. Each hashed feature and the list of target ontologies is transmitted to one node on the network. Each node on the network which receives a hashed feature uses the hashed feature of the query to perform a search on its respective partition of the database. The results of the searches of the local databases consist of the object identifiers of objects that match the query and the ontologies within which they were processed, as well as equivalent hashed features within other ontologies. These other hashed features are forwarded, as needed, to their respective nodes, and this process continues until the desired target ontologies are reached. When the target ontologies are reached, the results of the searches of the local databases are gathered by the home node. The results of the query are then computing for each target ontology. This process may be repeated by the home node to refine the results of the query.

References

[1] L. Aiello, J. Doyle, and S. Shapiro, editors. *Proc. Fifth Intern. Conf. on Principles of Knowledge Representation and Reasoning.* Morgan Kaufman Publishers, San Mateo, CA, 1996.

[2] K. Baclawski. Distributed computer database system and method, December 1997. United States Patent No. 5,694,593. Assigned to Northeastern University, Boston, MA.

[3] K. Baclawski and D. Simovici. An abstract model for semantically rich information retrieval. Technical report, Northestern University, Boston, MA, March 1994.

[4] A. Campbell and S. Shapiro. Algorithms for ontological mediation. Technical report, State University of New York at Buffalo, Buffalo, NY, 1998.

[5] A. Del Bimbo, editor. *The Ninth International Conference on Image Analysis and Processing*, volume 1311. Springer, September 1997.

[6] N. Fridman. *Knowledge Representation for Intelligent Information Retrieval in Experimental Sciences.* PhD thesis, College of Computer Science, Northeastern University, Boston, MA, 1997.

[7] R. Jain. Content-centric computing in visual systems. In *The Ninth International Conference on Image Analysis and Processing, Volume II*, pages 1–13, September 1997.

[8] Y. Ohta. *Knowledge-Based Interpretation of Outdoor Natural Color Scenes.* Pitman, Boston, MA, 1985.

[9] G. Salton. *Automatic Text Processing.* Addison-Wesley, Reading, MA, 1989.

[10] G. Salton, J. Allen, and C. Buckley. Automatic structuring and retrieval of large text files. *Comm. ACM*, 37(2):97–108, February 1994.

[11] A. Tversky. Features of similarity. *Psychological review*, 84(4):327–352, July 1977.

1 Field of the Invention

The invention relates to computer database systems and more specifically to distributed computer database systems.

2 Background of the Invention

The basis for communication whether it is between people or computer systems is a shared background that allows them to understand each other. This involves sharing both of the following:

1. A language for communication.

2. A domain conceptualization that defines the shared vocabulary along with relationships that may hold between the concepts denoted by the terms in the vocabulary.

The problem of translation between different languages is an important, and many computer systems have been developed for this purpose. However, translation between different domain conceptualizations is also important. Translation between domain conceptualizations is called *mediation*. Domain conceptualizations are also called *ontologies*.

For example, the vocabulary of Americans differs from that of the British even though they share a common language. In the UK, one would say "lift" for what is called an "elevator" in the US.

For a more complex example, the domain of medicine has a large vocabulary of terms for chemicals, genes, laboratory procedures, diseases, etc. Within medicine there are many subdomains that use different terminology for the same concept. Terminology can also vary from one company to another, and even small groups within a single company can have their own specialized vocabulary. Some will use the term "Munchausen Syndrome" while others prefer "Chronic factitious illness with physical symptoms". Some might even prefer to expand the term "factitious illness" to "intentional production or feigning of symptoms or disabilities, either physical or psychological" to make it understandable to someone with minimal medical background.

The problem of mediation between domain conceptualizations is especially difficult for computer systems because they generally have no mechanism for dealing with miscommunication as a result of misunderstood terminology. For example, modern search engines simply match words in a query with words in documents. Some search engines consider the possibility of synonymous words, but the fact that the words might belong to different domains is not considered.

For example, suppose that one wishes to find occurrences of "Job" in the Bible. Job is one of the persons mentioned in the Bible, and one of the books in the Bible is named after him. However, modern search engines do not generally understand this, and they will make errors such as matching "Job" with "work" because they regard these two words as synonymous.

Current search engines support only a very limited ontology with just a few concepts. Moreover, the ontology is inflexibly built into the search engine and only one ontology is supported. In general, indexes of current database systems are always limited to a single ontology.

A collection of documents, data or other kinds of information objects will be called an *object database*. Information objects can be images, sound and video streams, as well as data objects such as text files and structured documents. Each information object is identified uniquely by an object identifier (OID). An OID can be an Internet

Universal Resource Locator (URL) or some other form of identifier such as a local object identifier.

To assist in finding information in an object database, special search structures are employed called *indexes*. Current technology generally requires a separate index for each attribute or feature. Even the most sophisticated indexes currently available are limited to a very small number of attributes. Since each index can be as large as the database itself, this technology does not function well when there are hundreds or thousands of attributes, as is often the case when objects such as images, sound and video streams are directly indexed. Furthermore, there is considerable overhead associated with maintaining each index structure. This limits the number of attributes that can be indexed. Current systems are unable to scale up to support databases for which there are: many object types, including images, sound and video streams; millions of features; queries that involve many object types and features simultaneously; and new object types and features being continually added.

3 Summary of the Invention

In the present invention, each object in an object database is assumed to be expressed within a single domain. In other words, there is a particular domain conceptualization (ontology) that allows one to understand the object. However, different objects may be expressed within different ontologies and queries may use different ontologies from the documents being searched. In addition, a single ontology may contain one or more subontologies which may be used in this way by several ontologies. Since ontologies may be very large, it is important to allow them to be built from smaller components. In other words, subontologies constructed for one ontology may be *reused* by another.

An *ontology* models knowledge within a particular domain. An ontology can include a concept network, specialized vocabulary, syntactic forms and inference rules. In particular, an ontology specifies the features that objects can possess as well as how to extract features from objects. The extracted features are used for determining the degree of similarity between a query object and an object in the database. Each feature of an object may have an associated weight, representing the strength of the feature.

Ontologies can be related to one another in several ways, including the following:

Subset One ontology can be a subset of another. In this case the smaller ontology is called a *subontology* of the other. An ontology can be a subontology of several other ontologies.

Version One ontology can be a version of another. As domains evolve, concepts and terminology can change enough that it is necessary to construct a new ontology which does not simply contain the old ontology.

Parallel A single domain area can have different ontologies. For example, the same company could have different terminology within its development, manufacturing and marketing departments.

Except for the case of a subontology, an explicit *ontological mapping* or *ontological mediation* is required to express concepts in one ontology in terms of concepts in a related ontology.

It should be noted that the information objects themselves need not be stored in the database system itself so long as their locations are available in the database system. For example, each document in the World Wide Web is located using its Universal Resource Locator (URL).

Current technology commonly requires that information retrieval queries be specified in an artificial query language. In the present invention queries to retrieve objects in the database are in the same format as the objects in the database. While each object in the database conforms to exactly one ontology and each query conforms to exactly one ontology, a query can specify any number of target ontologies. For reasons of efficiency and performance the present invention does not have a step in which objects or queries are translated from one ontology to another ontology. The translation and the information retrieval are performed simultaneously in parallel over the distributed network of computer nodes, and objects conforming to all specified target ontologies are retrieved.

The invention relates to a distributed computer database system which includes one or more front end computers and one or more computer nodes interconnected by a network. The combination of computer nodes interconnected by a network operates as a search engine.

A user wishing to query the database, transmits the query, to one of the front end computers which in turn forwards the query to one of the computer nodes of the network. In the present invention, the notion of a query includes the ontology to be used for processing the query and the target ontologies of the objects to be retrieved. The node receiving the query, termed the home node of the search engine, extracts the features of the received query and then hashes these features. A portion of each hashed feature is used by the home node as an addressing index by which the home node transmits the hashed query feature to a node on the network. The list of target ontologies is transmitted along with the hashed query feature.

Each node on the network which receives a hashed query feature uses the hashed query feature to perform a search on its respective database. Nodes finding data corresponding to the hashed query feature return the OIDs of the objects possessing this feature and conforming to one of the target ontologies. Such OIDs are then gathered by the home node and a similarity function is computed based on the features that are in common with the query as well as the features that are in the query but not in the object. The similarity function is used to rank the objects. The OIDs

of the objects that have the largest similarity value are transmitted to the front end node.

If requested, higher levels of service may be provided. For level 2 or level 3 service, the OIDs obtained in the basic service above are transmitted to the nodes on the network by using a portion of each OID as an addressing index. In addition, if level 3 service is requested, the features each object has in common with the query are transmitted along with the OIDs to the same nodes on the network.

Each node on the network which receives an OID uses the OID to perform a search on its respective database for the corresponding object information. In level 2 service, auxiliary information is retrieved and transmitted to the front end node. The auxiliary information can include the URL of the object or an object summary or both.

For level 3 service, a dissimilarity value is computed based on the features that the object possesses but the query does not. The dissimilarity value as well as the auxiliary information about the object is transmitted to the home node. The dissimilarity values are gathered by the home node which uses them to modify the similarity values of the objects obtained in the first level of processing. The modified similarity values are used to rank the objects. The OIDs and any auxiliary information about the objects that have the largest similarity value are transmitted to the front end node.

Regardless of the level of service requested, the front end node formats the response to the user based on the OIDs and any auxiliary information transmitted by the home node. For example, if the front end node is a World Wide Web server, then the front end node constructs a page in HTML format containing a reference to a URL and auxiliary information for each object. The front end transmits the formatted response to the user.

4 Description of the Drawings

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

FIG. 1 is a block diagram of an overview of an embodiment of the distributed computer database system of the invention;

FIG. 2 is an overview of the steps used by the embodiment of the distributed computer database system to respond to a query;

FIG. 3 is an overview of the steps used by the embodiment of the distributed computer database system to store data associated with an information object;

FIG. 4 is an overview of the steps used by the embodiment of the distributed computer database system to store data associated with an ontology or associated with a mapping between ontologies.

FIG. 5 specifies the formats of the messages transmitted between the nodes of the distributed computer database system.

The remaining diagrams are block diagrams of the modules that perform the tasks of the invention within each node.

5 Detailed Description of the Preferred Embodiment

Referring to FIG. **1**, in broad overview, one embodiment of a distributed computer database system of the invention includes a user computer which is in communication with a front end computer through a network. The front end computer, which may also be the user computer, is in turn in communication with a search engine which includes one or more computer nodes interconnected by a local area network. The individual computer nodes may include local disks, or may, alternatively or additionally, obtain data from a network disk server.

The computer nodes of the search engine may be of several types, including home nodes, query nodes and object nodes. The nodes of the search engine need not represent distinct computers. In one embodiment, the search engine consists of a single computer which takes on the roles of all home nodes, query nodes and object nodes. In another embodiment, the search engine consists of separate computers for each home node, query node and object node. Those skilled in the art will realize many variations are possible which will still be within the scope and spirit of the present invention.

Considering the processing of a query first, and referring also to FIG. **2**, in one embodiment when a user transmits (Step **201**) a query from the user computer, the front end computer receives the query. The front end computer is responsible for establishing the connection with the user computer to enable the user to transmit a query and to receive a response in an appropriate format. The front end computer is also responsible for any authentication and administrative functionality. In one embodiment, the front end computer is a World Wide Web server communicating with the user computer using the HTTP protocol.

After verifying that the query is acceptable, the front end computer performs any reformatting necessary to make the query compatible with the requirements of the search engine. The front end computer then transmits the query to one of the home

nodes of the search engine (Step **202**), which is then defined as the home node of the search engine for that query.

The home node extracts features from the query according to the ontology. Feature extraction may be performed using traditional techniques for associating values to attributes, such as in relational database records. Note that an object either possesses a feature or it does not. This property is distinct from the value associated with a feature when an object possesses the feature.

Features are extracted from structured documents by parsing the document to produce a data structure, then dividing this data structure into (possibly overlapping) substructures called fragments. The fragments of a structured document are the features extracted from the document. Fragments of a query are used to find matching fragments in the database, so they are also called *probes*. This same terminology will be used for features extracted from other kinds of objects as well.

Features are extracted from unstructured documents by using knowledge extraction techniques. Knowledge extraction produces a data structure consisting of a collection of inter-related knowledge frames. The knowledge frame data structure is then divided into (possibly overlapping) substructures, as in the case of a structured document, and these substructures are the features of the unstructured document.

A large variety of feature extraction algorithms have been developed for media such as sound, images and video streams. Fourier and Wavelet transformations as well as many filtering algorithms are used. Features can also be added to an object by manual or semi-automated means. Such added features are referred to as annotations or meta-data. Features are extracted from annotations using one of the techniques mentioned above, depending on whether the annotation is a relational database record, a structured document or an unstructured document. Each feature can have a value associated with it, and one can specify relationships between features which can also have values associated with them. For example, one feature can be contained within another feature or be adjacent to another feature. The ontology specifies the feature extraction algorithms as well as the structure of the features.

If a feature occurs very commonly in the database, then it does not contribute to the purpose of the search engine; namely, distinguishing those objects that are similar to a particular query. An example is the brightness of an image. Such a feature will be partitioned into a collection of contiguous, non-overlapping ranges of the value associated with the feature rather than the feature itself. Each range of the value is then regarded as a separate feature. When the features of a query are extracted, features that represent value ranges near, but not including, the value of the feature in the query are also included as features of the query, but with smaller strength than the feature representing a value range that includes the value of the feature in the query. The value ranges for a particular feature can either be specified explicitly in the ontology, or they can be constructed dynamically as objects are indexed by the

search engine.

The home node then encodes each feature of the query by using a predefined hashing function. Data in the system was previously stored locally on the various query nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for a data query assures that 1.) data is distributed uniformly over the query nodes of the search engine during the storing of data and 2.) the probes are scattered uniformly over the query nodes during the processing of a query.

In one embodiment, the hash value resulting from the use of the hashing function has a first portion which serves to identify the query node to which the data is to be sent to be stored or to which a query feature is to be sent as a probe and a second portion which is the local index value which is used to determine where data is to be stored at or retrieved from the query node. Thus, in terms of a query, the hashed query features are distributed (Step **203**) as probes to certain query nodes of the search engine, as determined by the first portion of the hash value.

At a first or basic service level, query nodes whose probes match the index features by which the data was initially stored on that query node respond to the query in one or both of the following ways:

- If the ontology mapping of a hashed query fragment matches one of the specified ontology mappings, the hashed query fragment is transmitted (Step **204**) to the query node of the search engine, as determined by the first portion of the hash value. This query node repeats this process.

- If the ontology of an OID matches one of the target ontologies, the OID is transmitting (Step **205**) to the home node.

Thus all matches between the hashed probes, mapped to target ontologies, and the local hash table of index terms are returned or gathered to the home node which initially received the query.

The home node then determines the relevance of each object returned in the search. This determination of relevance is made by the home node by comparing the degree of similarity between the query and the objects whose OIDs were returned. The determination of relevance is made separately for the objects belonging to each target ontology. In one embodiment the measure of similarity between the query and the object is a cosine measure and is given by the expression $COS(v, w)$, where the vector v denotes the query and the vector w denotes the object. These vectors are in a space in which each feature represents one dimension of the space.

Another commonly used measure of similarity between two objects is a distance function in the same space mentioned above for the cosine measure. However, there

is convincing evidence that human similarity does not satisfy the axioms of a distance function. The model that currently seems to be the most successful approach is the Feature Contrast Model of Tversky. In this model, the similarity between a query and an object is determined by three terms:

1. The features that are common to the query and the object.

2. The features of the query that are not features of the object.

3. The features of the object that are not features of the query.

The first term contributes a positive number to the similarity value, while the second and third terms have negative contributions. In addition the second and third terms are multiplied by predefined constants such that a feature in the second and third set has less effect on the similarity than one in the first set.

In one embodiment the measure of similarity between the query and the object is a measure determined by three predefined constants that are used to multiply the three terms occurring in the Feature Contrast Model. In this embodiment, if the level of service is specified to be either basic or level **2**, then only the first two terms of the Feature Contrast Model are used to compute the measure of similarity, or equivalently, the predefined constant for the third term is set to zero. Since the third term is the least important, it has only a small effect on the ranking of the objects that are retrieved. If all three terms are to be used, then level **3** service can be requested.

In one embodiment the N objects with the highest similarity in each target ontology are returned. In another embodiment all objects which generate similarity values greater than a predetermined value are considered sufficiently similar to the query to be returned to the user as relevant information.

Once the similarity is determined, the home node orders the OIDs according to their degree of similarity in each ontology, and then returns a set of lists of the most relevant OIDs, each list containing the most relevant OIDs for one ontology. In one embodiment the set of lists of the most relevant OIDs is transmitted to the front end (Step **205**) computer which formats the response appropriately and transmits the response to the user. In another embodiment the set of lists of the most relevant OIDs is transmitted directly to the user computer by way of the network without the intervention of the front end computer.

Alternatively, for higher levels of service (level **2** and level **3**), the home node transmits the most relevant OIDs to the object nodes (Step **206**) which hold information associated with the objects identified by the OIDs. In one embodiment, the information associated with each object is the URL for the object. In another embodiment, the information associated with each object is the object itself. In another embodiment, the information associated with each object is the list of all features

of the object and the values of the features for those features that have associated values.

In one embodiment, the OIDs have a first portion which serves to identify the object node on which the object information is stored and a second portion which is the local index value which is used to determine where the object information is stored in a local table at the object node.

For level **2** service, the object nodes return the object information of the most relevant objects. In one embodiment the object information of the most relevant objects is transmitted to the front end (Step **207**) computer which formats the response appropriately and transmits the response to the user. In another embodiment the object information of the most relevant objects is transmitted directly to the user computer by way of the network without the intervention of the front end computer.

For level **3** service, the object nodes transmit the object information of the most relevant objects to the home node (Step **207**). The home node uses the object information of the relevant objects to recompute the measure of similarity between the query and the objects. This may result in the objects being arranged in a different order for each target ontology, and may also result in a different list of objects being returned for each target ontology. In one embodiment, the measure of similarity utilizes the Feature Contrast Model and all three terms have nonzero predefined constants. In this embodiment, the object information contains a list of the features of the object so that features of the object that are not features of the query may be included in the measure of similarity.

For level **3** service, the home node returns the object information of the most relevant objects. In one embodiment the object information of the most relevant objects is transmitted to the front end (Step **208**) computer which formats the response appropriately and transmits the response to the user. In another embodiment the object information of the most relevant objects is transmitted directly to the user computer by way of the network without the intervention of the front end computer.

Considering next the indexing of an object, and referring also to FIG. **3**, in one embodiment when a user transmits (Step **301**) an object from the user computer, the front end computer receives the object. The front end computer is responsible for establishing the connection with the user computer to enable the user to transmit an object. In another embodiment the front end computer automatically examines objects in its environment for indexing by the search engine without interaction with a user.

The front end selects a home node and transmits the object to the selected home node (Step **302**). In one embodiment, the selection of a home node is done randomly so as to evenly distribute the workload among the home nodes. The home node assigns a unique OID to the object, then processes the object as discussed above in the case of a query (Step **303**), except that data associated with the object is stored

in the query nodes and an object node.

Considering last the processing of an ontology mediation, and referring also to FIG. **4**, in one embodiment when a user transmits (Step **401**) an ontology mediation from the user computer, the front end computer receives the ontology mediation. The front end computer is responsible for establishing the connection with the user computer to enable the user to transmit an ontology mediation.

The front end selects a home node and transmits the ontology mediation to the selected home node (Step **302**). The home node constructs a feature mapping that implements the ontology mediation and transmits the feature mapping to every query node (Step **403**). Each query node stores the feature mapping in a local table. A background process is then started on each query node to examine every hashed object fragment and to store the mapped hashed object fragments if the feature mapping applies to it (Step **404**). This local table is consulted whenever a new hashed object fragment is stored as described above and referring also to FIG. **3**.

Considering next the message formats used in the preferred embodiment, refer to FIG. **5**. The Query Message has two parts: Identifier and Target. The Identifier part has four fields: Header, Query Identifier (QID), Hashed Query Fragment (HQF) and Value. The Header field specifies that this message is a Query Message and also specifies the destination query node. The destination query node is determined by the first portion of the hashed query fragment. The QID field contains a query type specifier and a query identifier. The query type specifier determines the ontology in which the query was specified. The HQF field contains a fragment type specifier and the second portion of the hashed query fragment produced by the Hashing Module. The Value field contains an optional value associated with the fragment. The fragment type specifier determines whether the Query Message contains a Value field, and if the Query Message does contain a Value field then the fragment type specifier determines the size of the Value field. The Target part contains a list of target ontology identifiers. An ontology is also called a *knowledge model*, and an ontology identifier is abbreviated KID.

The Query Response Message contains four fields: Header, QID, Object Identifier (OID) and Weight. The Header field specifies that this message is a Query Response Message and also specifies the destination home node. The destination home node is the home node from which the corresponding Query Message was received. The QID field contains a query type specifier and a query identifier. The OID field contains an object type specifier and an object identifier. The object type specifier determines the ontology in which the object was processed. The Weight field contains an optional weight associated with the object. The object type specifier determines whether the Query Response Message contains a Weight field, and if the Query Response Message does contain a Weight field then the object type specifier determines the size of the field.

The Object Message has three fields: Header, QID and OID. The Header field specifies that this message is an Object Message and also specifies the destination object node. The destination object node is determined by the first portion of the object identifier. The QID field contains a query type specifier and a query identifier. The OID field contains an object type specifier and the second portion of the object identifier.

The Object Response Message has three parts: Identifier, Feature and Auxiliary. The Identifier part has four fields: Header, QID, OID and Location. The Header field specifies that this message is an Object Response Message and also specifies the destination home node. The destination home node is the home node from which the corresponding Object Message was received. The QID field contains a query type specifier and a query identifier. The OID field contains an object type specifier and the object identifier. The Location field contains an optional location specifier such as a URL. The object type specifier determines whether the Object Response Message contains a Location field, and if the Object Response Message does contain a Location field, then the object type specifier determines the size of the Location field. The Feature part contains a number of features associated with the object. The Auxiliary part contains auxiliary information associated with the object. The object type specifier determines whether the Object Response Message contains an Auxiliary part, and if the Object Response Message does contain an Auxiliary part, then the object type specifier determines the size and structure of the Auxiliary part.

The Insert Message has two parts: Identifier and Target. The Identifier part has four fields: Header, OID, HQF and Value. The Header field specifies that this message is an Insert Message and also specifies the destination query node. The destination object node is determined by the first portion of the hashed query fragment. The OID field contains an object type specifier and the object identifier. The HQF field contains a fragment type specifier and the second portion of the hashed query fragment produced by the Hashing Module. The Value field contains an optional value associated with the fragment. The fragment type specifier determines whether the Query Message contains a Value field, and if the Query Message does contain a Value field then the fragment type specifier determines the size of the Value field. The Target part contains a list of target ontology identifiers.

The Insert Object Message has three parts: Identifier, Feature and Auxiliary. The Identifier part has three fields: Header, OID and Location. The Header field specifies that this message is an Insert Object Message and also specifies the destination object node. The destination object node is determined by the first portion of the object identifier. The OID field contains an object type specifier and the second portion of the object identifier. The Location field contains an optional location specifier such as a URL. The object type specifier determines whether the Insert Object Message contains a Location field, and if the Insert Object Message does contain a Location

field, then the object type specifier determines the size of the Location field. The Feature part contains a number of features associated with the object. The Auxiliary part contains auxiliary information associated with the object. The object type specifier determines whether the Insert Object Message contains an Auxiliary part, and if the Insert Object Message does contain an Auxiliary part, then the object type specifier determines the size and structure of the Auxiliary part.

The Mediation Message has two parts: Identifier and Mapping. The Identifier part has four fields: Header, MT, KID1 and KID2. The Header field specifies that this message is a Mediation Message and also specifies the destination query node. The Mediation Type (MT) field contains a mediation type specifier. The KID1 field contains the ontology identifier of the originating ontology from which the mediation takes place. The KID2 field contains the ontology identifier of the target ontology to which the mediation maps. The Mapping part contains the feature mapping specification. The structure and size of the feature mapping specification is determined by the MT field.

Considering next the Communication Module contained in the computer nodes used in the preferred embodiment, refer to Fig. 6, 7 and 8. The Communication Module is responsible for transmitting and receiving messages from one node to another. The destination node for a message to be transmitted is specified in the Header field of each message. When a message is received from another node, the type of message determines which module will process the message. The message type is specified in the Header field of each message.

The Communication Module of a home node is also responsible for communication with the Front End nodes. A Front End node transmits queries, objects and ontology mediations to the home node, and the home node transmits results to the Front End node.

Considering next the modules contained in the home nodes used in the preferred embodiment, refer to Fig. 6. The Feature Extractor extracts features from a query or object. Feature extraction for images is performed by detecting edges, identifying the image objects, classifying the image objects as domain objects and determining relationships between domain objects. In another embodiment, feature extraction for images is performed by computing Fourier or wavelet transforms. Each Fourier or wavelet transform constitutes one extracted feature. The Feature Extractor is also responsible for selecting target ontologies based on specifications in the query or object. The extracted features and target ontology identifiers are transferred to the Fragmenter. In addition, when features have been extracted from an object, the features are transferred to the Communication Module in the form of an Insert Object Message.

The Fragmenter computes the fragments contained in each feature. Each fragment consists of a bounded set of related components in the feature. In one embodiment,

the fragments of a feature consist of each attribute and each relationship in the data structure defining the feature. The fragments and target ontologies are transferred to the Hashing Module.

The Hashing Module computes a hash function of a fragment. In one embodiment, the hash function is the MD4 message digest function. The Hashing Module transfers either a Query Message or an Insert Message to the Communication Module, depending on whether the fragment is a query fragment or an object fragment, respectively.

The Similarity Comparator receives Query Response Messages and produces Object Messages which are transferred to the Communication Module. The Similarity Comparator gathers all the query responses for a query. For each object in the responses, the Similarity Comparator determines the relevance of each object returned in the search. This determination of relevance is made by the home node by comparing the degree of similarity between the query and the objects whose OIDs were returned. In one embodiment the measure of similarity between the query and the object is a cosine measure and is given by the expression $COS(v, w)$, where the vector v denotes the query and the vector w denotes the object. These vectors are in a space in which each fragment represents one dimension of the space. The most relevant OIDs are transferred to the Communication Module using an Object Message.

The Response Constructor receives Object Response Messages. It formats a response by collecting all the Object Response Messages having the same QID field. In one embodiment, each Object Response Messages results in one row of the formatted table. The entries in the row are determined by each feature of the Features part of the Object Response Message. In addition, one entry in the row specifies the Location field. The arrangement of the rows within the table is determined by the Auxiliary parts of the Object Request Messages. The formatted response is transmitted to the front end from which the query was received.

The Mediation Module receives ontology mediation specifications and transmits Mediation Messages. Ontology mediation maps terminology in one ontology, called the originating or source ontology, to terminology in another ontology, called the target or destination ontology. The Mediation Module determines the identifiers of the originating and target ontologies. The mediation specification determines a number of feature mappings. For example, "Munchausen Syndrome" in a medical ontology could be mapped to "Chronic intentional production of symptoms or disabilities, either physical or psychological" in another ontology. Each mediation specification is expressed as a feature mapping. There can be many types of mediation specification, such as definitions, ingredients and procedures. The mediation type determines the type of the mediation specification and the feature mapping. The Mediation Module constructs a Mediation Message for each query node. These Mediation Messages are identical except for the destination query node specified in the Header field. Each

Mediation Message is transferred to the Communication Module.

Considering next the modules contained in the query nodes used in the preferred embodiment, refer to Fig. 7. The Fragment Table receives Query Messages and Insert Messages. In the case of a Query Message the Fragment Table retrieves an entry in the local hash table using the hash value in the HQF field. The type specifier in the HQF field and the entry in the local hash table are transferred to the Fragment Comparator. If there are any target KIDs in the Query Message and if the entry specifies a mapping from the ontology of the entry to one of the target KIDs, then the Query Message and the entry are transferred to the Mapping Module.

In the case of an Insert Message, the Fragment Table modifies an entry in the local hash table by adding the OID and Value fields of the Insert Message to the entry in the local hash table. If there are any target KIDs in the Insert Message, then the Insert Message is transferred to the Mapping Module.

The Fragment Comparator receives entries from the Fragment Table. A comparison function is determined by the HQF type specifier that was transferred from the Fragment Table. The comparison function is used to determine the relevance of the OID and Value fields in the entry that was transferred from the Fragment Table. In one embodiment, the comparison function determines a similarity weight, and the OIDs having the highest similarity weight are deemed to be relevant. The relevant OIDs and their similarity weights are transferred to the Communication Module using a Query Response Message.

The Mapping Module receives Query Messages and Insert Messages from the Fragment Table. For a Query Message, the Mapping Module constructs a number of Query Messages for each target ontology specified by a target KID. When there is no direct mapping from the originating ontology to the target ontologies, the constructed Query Message will contain target KIDs. In this case the mapping proceeds through intermediate ontologies and requires several stages to reach the target ontology. For an Insert Message, the Mapping Module constructs a number of Insert Messages for each target ontology specified by a target KID, using the feature mapping in the local table of feature mappings.

The Mediation Module receives Mediation Messages. For each Mediation Message, a background process is initiated that is responsible for examining every hashed object fragment to determine the corresponding mapped hashed object fragments determined by the feature mapping. In addition, the feature mapping is stored in a local table which is used by the Mapping Module.

Considering next the module contained in the object nodes used in the preferred embodiment, refer to Fig. 8. The Object Table receives Object Messages and Insert Object Messages. In the case of an Object Message, the Object Table retrieves an entry in the local table using the object identifier in the OID field of the Object Message. The Object Message and the retrieved entry are transmitted to the Communication

Module using an Object Response Message. In the case of an Insert Object Message, the Object Table inserts a new entry in the local table. If an entry already exists for the specified object identifier, then the existing entry is replaced. The new or replacement entry contains the information in the Insert Object Message.

6 Claims

Having shown the preferred embodiment, those skilled in the art will realize many variations are possible which will still be within the scope and spirit of the claimed invention. Therefore, it is the intention to limit the invention only as indicated by the scope of the claims.

What is claimed is:

1. A method for information retrieval using fuzzy queries in a distributed computer database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

   (a) selecting a first one of said plurality of home nodes;

   (b) extracting, by said selected home node, a plurality of features and a plurality of target ontology identifiers from a query by a user;

   (c) hashing, by said selected home node, each said query feature of said plurality of query features, said hashed query feature having a first portion and a second portion;

   (d) transmitting, by said selected home node, each said hashed query feature of said plurality of query features and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query feature;

   (e) using by said query node, said second portion of said respective hashed query feature to access data according to a local hash table located on said query node;

   (f) using by said query node, said plurality of target ontology identifiers and said accessed data according to said respective hashed query feature, to extract a plurality of hashed features and a plurality of object identifiers;

   (g) transmitting, by said query node, each said hashed feature of said plurality of hashed features and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of each said hashed feature; and

(h) returning, by each said query node accessing data according to said respective hashed feature, a plurality of object identifiers corresponding to said accessed data to said selected home node.

2. The method of claim **1** further comprising the step of receiving, at said home node, said query from said user, prior to the step of extracting features from said query.

3. The method of claim **2** further comprising the steps of:

   (a) determining, by said home node, a measure of similarity between said accessed data and said query; and

   (b) returning to said user, by said home node, accessed data having a predetermined degree of similarity,

   subsequent to the step of returning said plurality of object identifiers.

4. The method of claim **3** wherein said measure of similarity is determined by a similarity function based on:

   (a) features possessed by both the said accessed data and the said query; and

   (b) features possessed only by the said query.

5. A method of storing objects or locations of objects in a manner which is conducive to information retrieval using fuzzy queries in a distributed computer database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

   (a) selecting a first one of said plurality of home nodes;

   (b) extracting, by said selected home node, a plurality of features from an object submitted by a user;

   (c) hashing, by said selected home node, each said object feature of said plurality of object features, said hashed object feature having a first portion and a second portion;

   (d) transmitting, by said selected home node, each said hashed object feature of said plurality of features to a respective one of said plurality of query nodes indicated by said first portion of each said hashed object feature;

   (e) using, by said query node, said second portion of said respective hashed object feature to store data according to a local hash table located on said query node; and

(f) applying, by said query node, any applicable ontology mappings to said hashed object feature and storing data in said local hash table located on said query node.

6. The method of claim **5** further comprising the step of receiving, at said home node, said object from said user, prior to the step of extracting features from said object.

7. A distributed computer database system having an information retrieval tool for handling queries from a user comprising:

   (a) a plurality of home nodes; and

   (b) a plurality of query nodes;

   (c) said plurality of home nodes and said plurality of query nodes connected by a network.

   (d) wherein each said home node, upon receiving a query from a user, extracts a plurality of features and a plurality of target ontology identifiers from said query, hashes each said query feature of said plurality of query features into a hashed query feature having a first portion and a second portion, and transmits each said hashed query feature and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed query feature,

   (e) wherein each said query node uses said second portion of said hashed query feature to access data according to a local hash table located on said query node,

   (f) further wherein each said query node uses said plurality of target ontology identifiers and said accessed data to extract a plurality of hashed features and a plurality of object identifiers, and transmits each said hashed feature of said plurality of hashed features and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed feature, and returns said plurality of object identifiers corresponding to said accessed data to said home node.

8. The distributed computer database system of claim **7** wherein said home node determines a measure of similarity between said accessed data and said query and returns to said user accessed data having a predetermined degree of similarity.

9. The method of claim **8** wherein said home node measures similarity using a similarity function determined by:

(a) features possessed by both the said accessed data and the said query; and

(b) features possessed only by the said query.

10. A distributed computer database system for storage and retrieval of information objects or locations of information objects, comprising

(a) a plurality of home nodes; and

(b) a plurality of query nodes;

(c) said plurality of home nodes and said plurality of query nodes connected by a network.

(d) wherein each said home node, upon receiving an object from a user, extracts a plurality of features from said object, hashes each said object feature of said plurality of object features into a hashed object feature having a first portion and a second portion, and transmits each said hashed object feature to a respective one of said plurality of query nodes indicated by said first portion of said hashed object feature, and

(e) wherein each said query node uses said second portion of said hashed object feature to store objects or locations of objects and hashed features defined by ontology mappings, according to a local hash table located on said query node.

11. A distributed computer database system having an information retrieval tool for handling queries from a user, comprising:

(a) a plurality of home nodes; and

(b) a plurality of query nodes;

(c) said plurality of home nodes and said plurality of query nodes connected by a network.

(d) each said home node, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

(e) a query task enqueued being resultant in, in response to a query command from said user, extracting a plurality of features and a plurality of target ontology identifiers from a query contained in said query command, hashing each said query feature of said plurality of query features into a hashed query feature having a first portion and a second portion, and transmitting a query message containing each said hashed query feature and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed query feature,

(f) said query node, upon receipt of said query message, using said second portion of said hashed query feature to access data according to a local hash table located on said query node, said data consisting of a plurality of object identifiers and a plurality of hashed features, transmitting a message returning a plurality of object identifiers corresponding to said accessed data to said home node, and for each hashed feature of said plurality of hashed features, transmitting a message containing said hashed feature and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed feature.

12. A distributed computer database system for storage and retrieval of information, comprising:

(a) a plurality of home nodes; and

(b) a plurality of query nodes;

(c) said plurality of home nodes and said plurality of query nodes connected by a network.

(d) each said home node, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

(e) an insert task enqueued, in response to an insert command from said user, extracting a plurality of features from an object contained in said insert command, hashing each said object feature of said plurality of object features into a hashed object feature having a first portion and a second portion, and transmitting an insert message containing each said hashed object feature to a respective one of said plurality of query nodes indicated by said first portion of said hashed object feature,

(f) said query node, upon receipt of said insert message, using said second portion of said hashed object feature to store data according to a local hash table located on said query node, and using any ontology mappings applicable to said hashed object feature to store data in said local hash table located on said query node.

13. A method for information retrieval using fuzzy queries in a distributed computer database system having a plurality of home nodes, a plurality of query nodes and a plurality of object nodes connected by a network, said method comprising the steps of:

(a) selecting a first one of said plurality of home nodes;

(b) extracting, by said selected home node, a plurality of features and a plurality of target ontology identifiers from a query by a user;

(c) hashing, by said selected home node, each said query feature of said plurality of query features, said hashed query feature having a first portion and a second portion;

(d) transmitting, by said selected home node, each said hashed query feature of said plurality of query features and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query feature;

(e) using by said query node, said second portion of said respective hashed query feature to access a plurality of object identifiers and a plurality of hashed features according to a local hash table located on said query node, each said object identifier having a first portion and a second portion;

(f) transmitting, by said query node, each said hashed feature of said plurality of hashed features and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of each said hashed feature;

(g) returning, by each said query node accessing data according to said respective hashed query feature, each said accessed object identifier to said selected home node.

(h) transmitting, by said selected home node, each said object identifier of said plurality of object identifiers to a respective one of said plurality of object nodes indicated by said first portion of each said object identifier;

(i) using by said object node, said second portion of said respective object node to access data according to a local object table located on said object node; and

(j) returning, by each said object node accessing data according to said respective object identifier, an object location, object features and other auxiliary information to said selected home node.

14. The method of claim **13** further comprising the step of receiving, at said home node, said query from said user, prior to the step of extracting features from said query.

15. The method of claim **14** further comprising the steps of:

(a) determining, by said home node, a measure of similarity between said accessed data and said query; and

(b) returning to said user, by said home node, accessed data having a predetermined degree of similarity,

subsequent to the step of returning said object location and auxiliary data.

16. The method of claim **15** wherein said measure of similarity is determined by a similarity function based on:

(a) features possessed by both the said accessed data and the said query;

(b) features possessed only by the said query; and

(c) features possessed only by said accessed data.

17. A method of storing objects or locations of objects in a manner which is conducive to information retrieval using fuzzy queries in a distributed computer database system having a plurality of home nodes, a plurality of object nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

(a) selecting a first one of said plurality of home nodes;

(b) selecting, by said selected home node, a unique object identifier for an object selected by a user, said object identifier having a first portion and a second portion;

(c) using the said first portion of said object identifier to select one of said plurality of object nodes;

(d) extracting, by said selected home node, a plurality of features from the said object submitted by a user.

(e) hashing, by said selected home node, each said object feature of said plurality of object features, said hashed object feature having a first portion and a second portion;

(f) transmitting, by said selected home node, the location of the said object, the said plurality of hashed object features of the said object and any auxiliary information about the said object to a respective one of said plurality of object nodes indicated by said first portion of said object identifier;

(g) using, by said object node, said second portion of said object identifier to store data according to a local object table located on said object node;

(h) transmitting, by said selected home node, each said hashed object feature of said plurality of features to a respective one of said plurality of query nodes indicated by said first portion of each said hashed object feature;

(i) using, by said query node, said second portion of said respective hashed object feature to store data according to a local hash table located on said query node; and

(j) applying, by said query node, any applicable ontology mappings to said hashed object feature and storing data in said local hash table located on said query node.

18. The method of claim **17** further comprising the step of receiving, at said object node, said object from said user, prior to the step of extracting features from said object.

19. A distributed computer database system having an information retrieval tool for handling queries from a user comprising:

(a) a plurality of home nodes;

(b) a plurality of query nodes; and

(c) a plurality of object nodes;

(d) said plurality of home nodes, said plurality of query nodes, and said plurality of object nodes connected by a network.

(e) wherein each said home node, upon receiving a query from a user, extracts a plurality of features and a plurality of target ontology identifiers from said query, hashes each said query feature of said plurality of query features into a hashed query feature having a first portion and a second portion, and transmits each said hashed query feature and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed query feature,

(f) further wherein each said query node uses said second portion of said hashed query feature to access data according to a local hash table located on said query node, said data consisting of a plurality of object identifiers and a plurality of hashed features, returns said plurality of object identifiers to said home node and transmits said each hashed feature of said plurality of hashed features and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of each said hashed feature,

(g) further wherein the said home node, upon receiving a plurality of object identifiers from each said query node, divides each said object identifier of said plurality of object identifiers into a first portion and a second portion, and transmits each said object identifier to a respective one of said plurality of object nodes indicated by said first portion of said object identifier, and

(h) further wherein each said object node uses said second portion of said object identifier to access data according to a local object table located on said object node and returns said accessed data to said home node.

20. The distributed computer database system of claim **19** wherein said home node determines a measure of similarity between said accessed data and said query and returns to said user accessed data having a predetermined degree of similarity.

21. The method of claim **20** wherein said home node measures similarity using a similarity function determined by:

(a) features possessed by both the said accessed data and the said query;

(b) features possessed only by the said query; and

(c) features possessed only by said accessed data.

22. A distributed computer database system for storage and retrieval of information objects or locations of information objects, comprising

(a) a plurality of home nodes;

(b) a plurality of object nodes; and

(c) a plurality of query nodes;

(d) said plurality of home nodes, said plurality of object nodes and said plurality of query nodes connected by a network.

(e) wherein each said home node, upon receiving an object from a user, selects a unique object identifier having a first portion and a second portion, extracts a plurality of features from said object, hashes each said object feature of said plurality of object features into a hashed object feature having a first portion and a second portion, transmits the location of the said object, the said plurality of hashed object features of the said object and any auxiliary information about the said object, to a respective one of said plurality of object nodes indicated by said first portion of said object identifier, and transmits each said hashed object feature to a respective one of said plurality of query nodes indicated by said first portion of said hashed object feature,

(f) wherein the said object node uses said second portion of said object identifier to store the location of the said object, the said plurality of hashed object features and auxiliary information about the said object, according to a local table located on said object node, and

(g) wherein each said query node uses said second portion of said hashed object feature to store objects or locations of objects and hashed features defined by ontology mappings, according to a local hash table located on said query node.

23. A distributed computer database system having an information retrieval tool for handling queries from a user, comprising:

(a) a plurality of home nodes;

(b) a plurality of query nodes; and

(c) a plurality of object nodes;

(d) said plurality of home nodes, said plurality of query nodes and said plurality of object nodes connected by a network.

(e) each said home node, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

(f) a query task enqueued being resultant in, in response to a query command from said user, extracting a plurality of features and a plurality of target ontology identifiers from a query contained in said query command, hashing each said query feature of said plurality of query features into a hashed query feature having a first portion and a second portion, and transmitting a query message containing each said hashed query feature and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed query feature,

(g) said query node, upon receipt of said query message, using said second portion of said hashed query feature to access data, consisting of a plurality of hashed features and a plurality of object identifiers, each said object identifier having a first portion and a second portion, according to a local hash table located on said query node, transmitting a message returning the said plurality of object identifiers corresponding to said accessed data to said home node, and for each hashed feature of said plurality of hashed features, transmitting a message containing said hashed feature and said plurality of target ontology identifiers to a respective one of said plurality of query nodes indicated by said first portion of said hashed feature.

(h) said home node, upon receipt of said message containing said plurality of object identifiers, transmitting an object message containing each said object identifier to a respective one of said plurality of object nodes indicated by said first portion of said object identifier,

(i) said object node, upon receipt of said object message, using said second portion of said object identifier to access data according to a local object

table located on said object node and transmitting a message returning the location of the object, the plurality of hashed object features of said said object and auxiliary information associated with the said object, corresponding to said accessed data to said home node.

24. The method of claim **23** wherein said query message requests predetermined data from said query node in response to a query and query level contained in said query command from said user.

25. The method of claim **24** wherein there are two query levels.

26. The method of claim **25** wherein said query node returns a plurality of locations of objects and auxiliary data in response to a predetermined query level.

27. A distributed computer database system for storage and retrieval of information, comprising:

    (a) a plurality of home nodes;

    (b) a plurality of object nodes; and

    (c) a plurality of query nodes;

    (d) said plurality of home nodes, said plurality of object nodes and said plurality of query nodes connected by a network.

    (e) each said home node, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

    (f) an insert task enqueued, in response to an insert command from said user, selecting a unique object identifier having a first portion and a second portion, for the object contained in said insert command, extracting a plurality of features from said object, hashing each said object feature of said plurality of object features into a hashed object feature having a first portion and a second portion, transmitting an object message containing the location of the said object, the said plurality of hashed object features and auxiliary information about the said object to a respective one of said plurality of object nodes indicated by said first portion of said object identifier, and transmitting an insert message containing each said hashed object feature to a respective one of said plurality of query nodes indicated by said first portion of said hashed object feature,

    (g) said object node, upon receipt of said object message, using said second portion of said object identifier to store data according to a local table located on said object node,

(h) said query node, upon receipt of said insert message, using said second portion of said hashed object feature to store data according to a local hash table located on said query node, and using any ontology mappings applicable to said hashed object feature to store data in said local hash table located on said query node.

28. A method of storing ontology mappings in a manner which is conducive to information retrieval using fuzzy queries in a distributed computer database system having a plurality of home nodes and a plurality of query nodes connected by a network, said method comprising the steps of:

(a) selecting a first one of said plurality of home nodes;

(b) transmitting, by said selected home node, said ontology mapping to said plurality of query nodes;

(c) using, by said query node, said ontology mapping to map all features to which said ontology mapping is applicable and which are stored in a local hash table located on said query node; and

(d) storing said ontology mapping in a local ontology mapping table located on said query node.

29. The method of claim **28** further comprising the step of receiving, at said home node, said ontology mapping from said user, prior to the step of transmitting said ontology mapping to said home node.

30. A distributed computer database system for storage and retrieval of information objects or locations of information objects, comprising

(a) a plurality of home nodes; and

(b) a plurality of query nodes;

(c) said plurality of home nodes and said plurality of query nodes connected by a network.

(d) wherein each said home node, upon receiving an ontology mapping from a user, transmits said ontology mapping to a respective one of said plurality of query nodes, and

(e) wherein each said query node uses said ontology mapping to map all features to which said ontology mapping is applicable and which are stored in a local hash table located on said query node, and to store said ontology mapping in a local ontology mapping table located on said query node.

31. A distributed computer database system for storage and retrieval of information, comprising:

    (a) a plurality of home nodes; and

    (b) a plurality of query nodes;

    (c) said plurality of home nodes and said plurality of query nodes connected by a network.

    (d) each said home node, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

    (e) an ontology mapping task enqueued, in response to an ontology mapping command from said user, transmitting an ontology mapping message to said plurality of query nodes,

    (f) said query node, upon receipt of said ontology mapping message, mapping all features to which said ontology mapping is applicable and which are stored in a local hash table located on said query node, and storing said ontology mapping in a local ontology mapping table located on said query node.

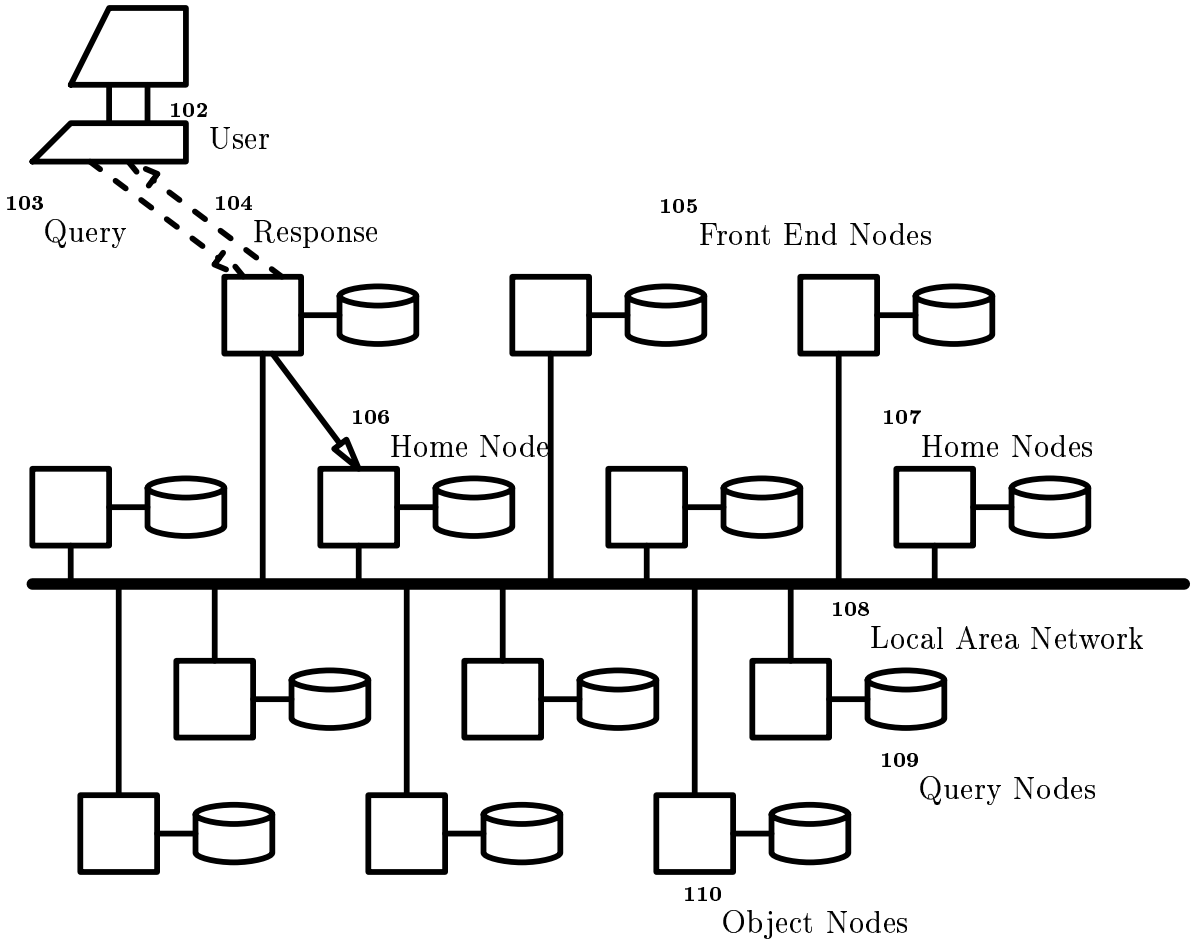
102
User
103
Query
104
Response
105
Front End Nodes
106
Home Node
107
Home Nodes
108
Local Area Network
109
Query Nodes
110
Object Nodes

FIG. **1**

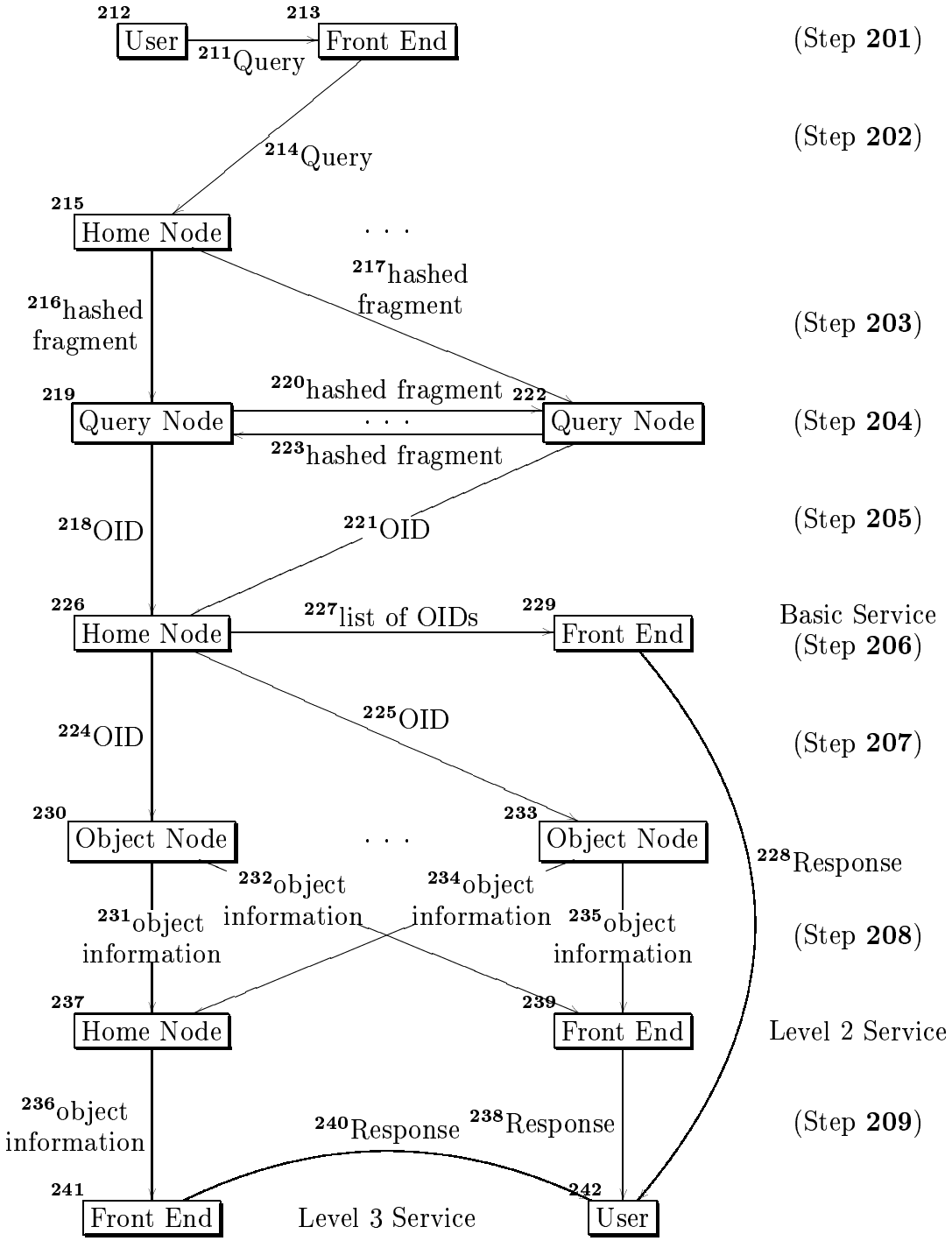
212
User
213
Front End
211Query
(Step 201)
214Query
(Step 202)
215
Home Node
. . .
217hashed fragment
216hashed fragment
(Step 203)
219
Query Node
220hashed fragment
222
Query Node
. . .
223hashed fragment
(Step 204)
218OID
221OID
(Step 205)
226
Home Node
227list of OIDs
229
Front End
Basic Service (Step 206)
225OID
224OID
(Step 207)
230
Object Node
. . .
233
Object Node
228Response
232object information
234object information
231object information
235object information
(Step 208)
237
Home Node
239
Front End
Level 2 Service
236object information
240Response
238Response
(Step 209)
241
Front End
Level 3 Service
242
User

FIG. **2**

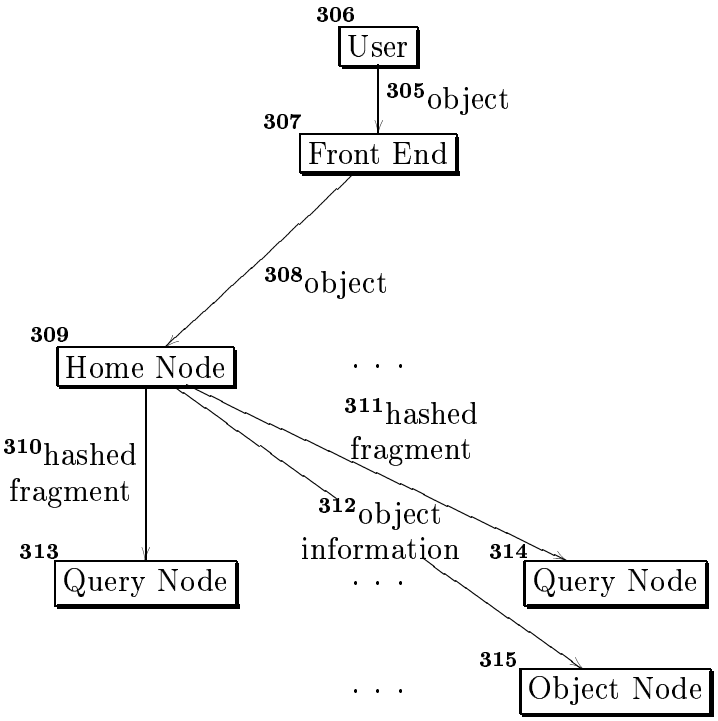
306
User
(Step 301)
305 object
307
Front End
(Step 302)
308 object
309
Home Node
. . .
311 hashed fragment
310 hashed fragment
(Step 303)
312 object information
313
Query Node
. . .
314
Query Node
315
Object Node
. . .

FIG. **3**

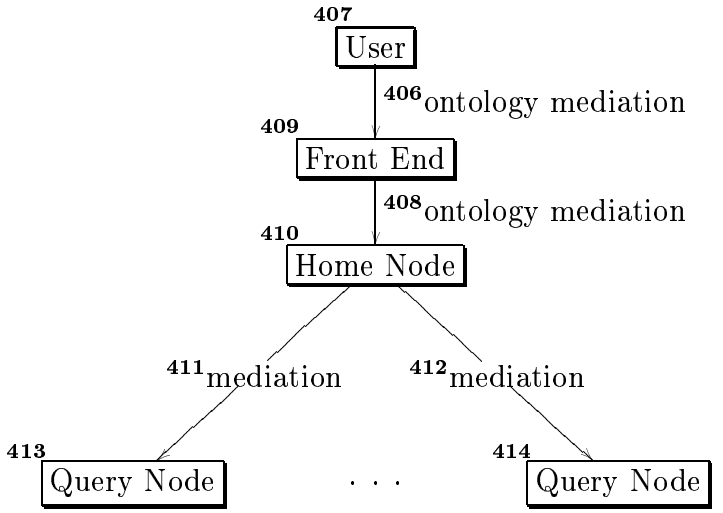
407
User
(Step 401)
406 ontology mediation
409
Front End
(Step 402)
408 ontology mediation
410
Home Node
411 mediation
412 mediation
(Step 403)
413
Query Node
. . .
414
Query Node
(Step 404)

FIG. **4**

Query Message

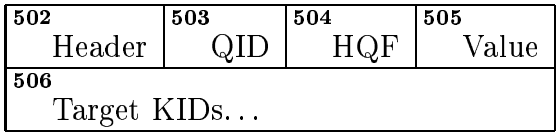

FIG. **5a**

Query Response Message

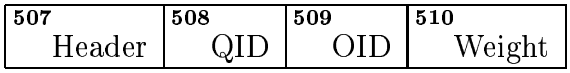

FIG. **5b**

Object Message

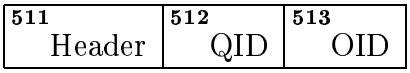

FIG. **5c**

Object Response Message

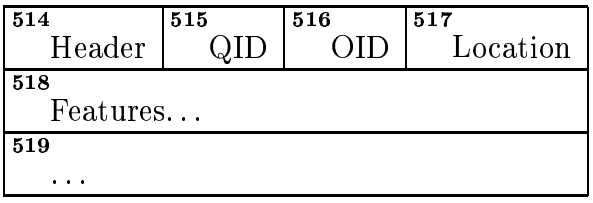

FIG. **5d**

Insert Message

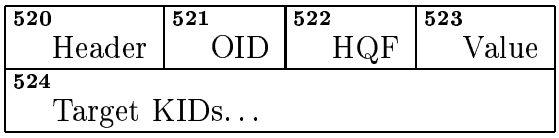

FIG. **5e**

Insert Object Message

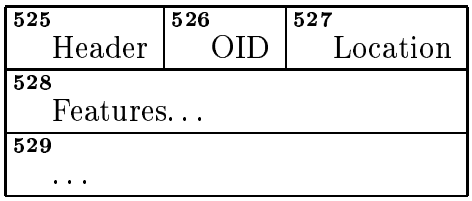

FIG. **5f**

Mediation Message

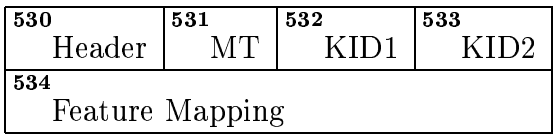

FIG. **5g**