Hypertext Linkage Analysis System and Method

Kenneth Baclawski, Waltham, MA

Abstract

A distributed computer database system including one or more front end computers, one or more home nodes, one or more index nodes and one or more object nodes interconnected by a network into a search engine for retrieval of hypertext documents. A query is a combination of one or more elementary queries. An elementary query is an index query, a link query or an object query. An index query is an object in the same format as the objects to be retrieved. A link query is an object in the same format as an index query except that it contains a specification for one incoming or outgoing hypertext link. An object query is a request for information about one object that is indexed by the search engine. A query consists of a series of elementary queries. An elementary query can make use of information retrieved by other elementary queries. 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 parses the query into elementary queries and schedules the elementary queries for processing. To process an index query or link query, the home node extracts features from the index query or link query and hashes these features. Each hashed feature is transmitted to one index node on the network. Each index node on the network which receives a hashed feature uses the hashed feature of the index query or link query to perform a search on its respective partition of the database. The results of the searches of the local databases are gathered by the home node. To process an object query, the home node transmits the object identifier contained in the object query to the object node on the network containing the information associated with the object. The object node which receives the object query uses the object identifier to perform a search on its respective partition of the database. The results of the search of the local database is transmitted to the home node. The home node processes the results of each elementary query according to the specifications in the query. The processing may include the evaluation of additional elementary queries. When all processing is completed by the home node, the results are returned to the front end node which formats the results for presentation to the user.

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] G. Arocena, A. Mendelzon, and G. Mihaila. Applications of a web query language. In *Proc. 6^{th} Intern. World Wide Web Conf.*, 1997.

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

[4] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proc. 7^{th} Intern. World Wide Web Conf.*, 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] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring Web communities from link topology. In *Proc. 9^{th} ACM Conf. on Hypertext and Hypermedia*, 1998.

[8] 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.

[9] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. ACM-SIAM Sympos. on Discrete Algorithms*, 1998.

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

[11] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. In *CHI'96 Proceedings: Conference on Human Factors in Computing Systems: Common Ground*, pages 118–125, Vancouver, BC, 1996.

[12] E. Rivlin, R. Botafogo, and B. Schneiderman. Navigating in hyperspace: Designing a structure-based toolbox. *Comm. of the ACM*, 37(2):87–96, February 1994.

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

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

[15] E. Spertus. ParaSite: Mining structural information on the web. In *Proc. 6th Intern. World Wide Web Conf.*, 1997.

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

[17] R. Weiss, B. Velez, M. Sheldon, C. Nemprempre, P. Szilagyi, and C. Giffor. HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proc. Seventh ACM Conf. on Hypertext*, pages 180–193, 1996.

[18] H. White and K. McCain. Bibliometrics. *Ann. Rev. Info. Sci. and Technology*, pages 119–186, 1989.

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 World Wide Web (WWW) is much more than just a collection of Web pages. Each page contains references to other pages. Such references are called *links*, and one of the most important features of a Web browser is the ability to follow a link and display the page that is being referenced. A collection of documents linked together in this way is called a *hypertext*.

The link structure of a hypertext is a rich source of knowledge about the content of the hypertext. In the field of bibliometrics, links in the form of citations have been used for understanding documents by using citation analysis techniques. The link structure of the WWW is now being exploited as a means of categorization and knowledge extraction. This is being done in two ways:

1. General hypertext query languages.

2. Cluster analysis algorithms.

A Web query language, such as WebSQL, is a query language for extracting information from the Web, based on hypertext structure as well as content. For example, one might be interested in a job opportunity for a librarian. One can query the Web using WebSQL to find all pages containing the keywords "employment" or "job opportunities" and then list all the pages referenced by such a page and containing the keyword "librarian."

Cluster analysis algorithms make use of Web query languages to find specific patterns in the link structure of the WWW. The most common cluster analysis pattern is the *authority/hub* pattern. To compute this pattern, one first specifies a topic area using one or more keywords. For example, one might be interested in the topic "Y2K." A page is potentially relevant if it contains one or more keywords of the topic. An *authority* page for a topic is a page that is referenced by a large number of pages potentially relevant to the topic. Note that an authority page need not contain any of the keywords of the topic. Authority is *conferred* on it by virtue of being referenced frequently by potentially relevant pages. A *hub* page for a topic is one that references a large number of pages potentially relevant to the topic. An authority page for "Y2K" is one that is highly referenced by pages that mention "Y2K." If one is interested in the Y2K problem, then it seems natural to look first at the authority pages.

Web query languages in general, and Web cluster analysis algorithms in particular, are limited in an important respect. They can only evaluate outgoing links, not incoming links. This is due to the way that Web links are defined. A link within one page specifies the link to which it linked, not the other way around. For example, suppose that one was interested in all the pages that refer to one's own home page. WebSQL cannot answer such a query.

The WWW is not just a hypertext. Pages can contain images, sound and video streams, and the structure of the WWW is continually changing. For these reasons, the WWW is called a *hypermedia environment*. Web resources are located by a Universal Resource Locator (URL) which uniquely identifies the resource. More generally, a hypermedia environment consists of information objects that are uniquely identified by an object identifier (OID) and that can contain links to other information objects. A hypermedia environment is also called an *object database*.

To assist in finding information in an object database, special search structures are employed called *indexes*. Large databases require correspondingly large index structures to maintain pointers to the stored data. Such an index structure can be larger than the database itself. Current technology requires a separate index for each attribute or feature. This technology can be extended to allow for indexing a small number of attributes or features in a single index structure, but this technology does not function well when there are hundreds or thousands of attributes. Furthermore, there is considerable overhead associated with maintaining an index structure. This limits the number of attributes or features that can be indexed. Current systems are

unable to scale up to support databases for which there are: many object types; 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

The present invention is an indexing and search engine that supports arbitrary query languages for extraction of information based on the content of the information objects in the database as well as the links between information objects. Unlike Web query languages such as WebSQL, the present invention fully supports queries involving either outgoing or incoming links. For example, the present invention can determine all the pages that refer to one's own home page.

Indexing of an object database requires an ontology. An *ontology* consists of a series of specifications that define the nature of being and the kinds of existence for the particular domain represented by the database. Most ontologies are very limited. For a relational database, the ontology is specified by its database schema which consists of the attributes of the records along with the types of the attribute values. Ontologies in general may include some or all of the following: vocabulary terms and term recognizers, conceptual categories, category classifications, relationships between conceptual categories, weight information that determines the strength of a relationship, syntactic forms for expressing these relationships, and logical inferences for relationships. In particular, an ontology specifies the features that information objects can possess as well as how to extract features from information objects. Each feature of an information object may have an associated weight, representing the strength of the feature.

Hypertext query languages and algorithms that make use of them, such as cluster algorithms, depend on the retrieval of three kinds of information in an object database:

1. Retrieval of objects relevant to a query. This is the traditional information retrieval problem.

2. Retrieval of link information relevant to a query.

3. Retrieval of all link information for a specific object. This includes both incoming and outgoing links.

A general query is composed of several *elementary queries* corresponding to the three kinds of retrieval described above:

index query An elementary query for retrieval of relevant objects.

link query An elementary query for retrieval of relevant link information.

object query An elementary query for retrieval of link information for one object.

The invention relates to a distributed computer database system which includes one or more front end computers, one or more home nodes, one or more index nodes and one or more object nodes interconnected by a network. A single computer processor can fulfill the functionality of one or more front end, home, index and object nodes. 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 nodes which in turn forwards the query to one of the home nodes of the network. The node receiving the query, termed the home node of this query, parses the query into elementary queries.

For an index or link query, the home node extracts the features of the received query and then encodes the features using a hash function. A portion of each hashed feature is used by the home node as an addressing index by which the home node transmits the hashed index or link query feature to an index node on the network. For an object query, the home node uses a portion of the OID as an addressing index by which the home node transmits the object query to an object node on the network.

Each index node on the network which receives a hashed index or link query feature uses the hashed index or link query feature to perform a search on its respective database. Index nodes finding data corresponding to a hashed index query feature return the set of OIDs of the information objects possessing this feature. Index nodes finding data corresponding to a hashed link query feature return the set of pairs of OIDs of the links between information objects which possess this feature.

Each object node on the network which receives an object query uses the OID contained in the object query to perform a search on its respective database. The object node returns the information associated with the OID as specified in the object query. Such information may include any or all of the following: the location of the object whose OID is contained in the object query, the set of OIDs that represent objects referenced by the object whose OID is contained in the object query, the set of OIDs that represent objects that reference the object whose OID is contained in the object query, and other auxiliary information associated with the object whose OID is contained in the object query.

The OIDs or pairs of OIDs are then gathered by the home node. For an index or link query, a similarity function is computed based on the features that are in common with the index or link query. The similarity function is used to rank the objects or links between objects. The objects or links between objects that have the largest similarity value are used in subsequent processing of the query. For an object query, the information returned by the object node is used in subsequent processing of the query.

The subsequent processing of the query by the home node may involve the construction of new elementary queries using the information returned from earlier elementary queries. Processing continues until no additional elementary queries are needed. The home node then performs any remaining processing required by the query, and the results are transmitted to the front-end node.

The front end node formats the response to the user based on the OIDs and any other 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 a hypertext document.

FIG. 4 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, index 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, index nodes and object nodes. In another embodiment, the search engine consists of separate computers for each home node, index 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 parses the query into a series of elementary queries. Elementary queries are of three kinds: index queries, link queries and object queries. Each elementary query consists of a command that determines what action the elementary query is to perform and additional information that depends on the kind of elementary query. An object query contains an OID, and the other two kinds of elementary query contain an information object conforming to the ontology of the object database. A link query contains exactly one link to an unspecified information object. An index query does not contain any unspecified links to other information objects.

The purpose of an object query is to obtain information about the object identified by the OID of the object query. The purpose of an index query is to search for information objects that contain information similar to the information in the index query. The purpose of a link query is to search for pairs of information objects such that one object contains information similar to the information in the link query and such that this information is associated with a link to the other information object.

The home node extracts information from each elementary query depending on what kind of elementary query it is. The home node extracts the OID from an object

query. The home node extracts features from an index query or a link query according to the ontology. 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 elementary queries or documents by parsing the document to produce a data structure, then dividing this data structure into (possibly overlapping) substructures called fragments. Fragments of an elementary query feature are used to find matching fragments in the database, so they are also called *probes*.

Features are extracted from unstructured elementary queries or documents by using feature extraction algorithms. Feature extraction produces a data structure consisting of a collection of inter-related domain objects. The data structure is divided into (possibly overlapping) substructures, as in the case of a structured document, and these substructures are the fragments of the unstructured document.

A large variety of feature extraction algorithms have been developed for media such as sound, images and video streams, such as edge detection, segmentation and object classification algorithms. Fourier and Wavelet transformations as well as many filtering algorithms are also used to extract features. Each feature can have one or more values associated with components of the data structure that represents the feature. In the simplest case the data structure consists of a single component with an associated value. In this case the feature represents one attribute of the object. More complex features will contain several inter-related components, each of which may have attribute values. For example, components of a feature can be contained within other components of the feature or be adjacent to other components.

The data structures that represent the features conform to a data model specified by the ontology. The data model determines the kinds of components and attribute values that are allowed. Each fragment of each feature has an associated weight, representing the strength of the feature.

If a fragment 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 fragment will be partitioned into a collection of contiguous, non-overlapping ranges of the value associated with the fragment rather than the fragment itself. Each range of the value is then regarded as a separate fragment. When the fragments of a query are extracted, fragments that represent value ranges near, but not including, the value of the fragment in the query are also included as fragments of the query, but with smaller strength than the fragment representing a value range that includes the value of the fragment in the query. The value ranges for a particular fragment 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 encodes each fragment of the query by using a predefined hashing function. Data in the system was previously stored locally on the various index nodes using this hashing function to generate an index to the data in the local database. In particular, if a fragment includes a link then it is hashed and stored as a link fragment, while if a fragment does not include a link then it is hashed and stored as an index fragment. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for a query assures that

1. data is distributed uniformly over the index nodes of the search engine during the storing of data and

2. the probes are scattered uniformly over the index 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 index 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 index node. Thus, in terms of an index or link query, the hashed query features are distributed (Step **203**) as probes to certain index nodes of the search engine, as determined by the first portion of the hash value.

In one embodiment, the OID of an object query has a first portion which serves to identify the object node to which the data is to be sent to be stored or to which an OID 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 object node (Step **203**).

Index nodes whose probes match the index features by which the data was initially stored on that index node respond to the elementary query by transmitting (Step **204**) the OIDs matching the index terms of the requested information to the home node. Thus all matches between the hashed probes and the local hash table of index terms are returned or gathered to the home node which initially transmitted the elementary query.

Index nodes whose probes match the link features by which the data was initially stored on that index node respond to the elementary query by transmitting (Step **204**) a set of pairs of OIDs matching the link terms of the requested information to the home node. Each pair of OIDs represents a link from the first OID in the pair to the second OID in the pair. Thus all matches between the hashed probes and the local hash table of link terms are returned or gathered to the home node which initially transmitted the elementary query.

Object nodes whose probes match the OIDs by which the data was initially stored on that object node respond to the object query by transmitting (Step **204**) information about the object and a set of OIDs associated with OID of the requested object

query to the home node. Each transmitted OID represents an object for which there is an incoming or outgoing link with the OID of the object query.

The home node then processes the results of each elementary query. The action is determined by the command contained in the elementary query. For an object query, the command may require selecting a subset of the OIDs according to criteria such as the directionality or type of the link.

For an index or link query, the command may require that the information returned to the home node be ranked and selected according to their relevance. 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 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 two predefined constants that are used to multiply the first two terms occurring in the Feature Contrast Model. In this embodiment, the predefined constant for the third term is assumed to be zero. Since the third term is the least important, it has only a small effect on the ranking of the objects that are retrieved.

In one embodiment the N objects with the highest similarity 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 objects according to their degree of similarity and then determines a list of the most relevant objects.

The result of an elementary query depends on the kind of elementary query. For an index query the result is a list of OIDs. For a link query the result is a list of pairs of OIDs. For an object query the result is object information, possibly including a list of OIDs.

The result list of an elementary query may be used in two ways depending on how the elementary query was related to other elementary queries when the original query was parsed. The result may be used for subsequent processing of other elementary queries, or it may be used as part of the information to be returned, or both. If it is used for subsequent processing of other elementary queries, these other elementary queries are processed as described above.

When all elementary queries have been processed, the results are collected for return to the user. In one embodiment the returned information is transmitted to the front end (Step **205**) computer which formats the response appropriately and transmits the response to the user (Step **206**). In another embodiment the information to be returned 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 elementary queries (Step **303**), except that data associated with the object is stored in the index nodes and an object node.

Considering next the message formats used in the preferred embodiment, refer to FIG. **4**. The Index Query Message has four fields: Header, Elementary Query Identifier (EQID), Hashed Query Fragment (HQF) and Value. The Header field specifies that this message is an Index Query Message and also specifies the destination index node. The destination index node is determined by the first portion of the hashed query fragment. The EQID field contains an elementary query type specifier and an elementary query 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 Index Query Message contains a Value field, and if the Index Query Message does contain a Value field then the fragment type specifier determines the size of the Value field.

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

The Link Query Message has four fields: Header, EQID, HQF and Value. The Header field specifies that this message is a Link Query Message and also specifies the destination index node. The destination index node is determined by the first portion of the hashed query fragment. The EQID field contains an elementary query type specifier and an elementary query 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 Link Query Message contains a Value field, and if the Link Query Message does contain a Value field then the fragment type specifier determines the size of the Value field.

The Link Query Response Message contains five fields: Header, EQID, OID1, OID2 and Weight. The Header field specifies that this message is an Link Query Response Message and also specifies the destination home node. The destination home node is the home node from which the corresponding Link Query Message was received. The EQID field contains an elementary query type specifier and an elementary query identifier. The two OID fields contain an object type specifier and an object identifier. The OID1 field contains the OID of the originating (source) object of the link. The OID2 field contains the OID of the destination (target) object of the link. The Weight field contains an optional weight associated with the object. The object type specifier of OID1 determines whether the Link Query Response Message contains a Weight field, and if the Link Query Response Message does contain a Weight field then the object type specifier determines the size of the field.

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

and the second portion of the object identifier.

The Object Query Response Message has three parts: Identifier, Feature and Auxiliary. The Identifier part has four fields: Header, EQID, OID and Location. The Header field specifies that this message is an Object Query Response Message and also specifies the destination home node. The destination home node is the home node from which the corresponding Object Query Message was received. The EQID field contains an elementary query type specifier and an elementary 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 Query Response Message contains a Location field, and if the Object Query 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 Index Message has four fields: Header, OID, HQF and Value. The Header field specifies that this message is an Insert Index Message and also specifies the destination index node. The destination index 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 Insert Link Message has five fields: Header, OID1, OID2, HQF and Value. The Header field specifies that this message is an Insert Link Message and also specifies the destination index node. The destination index node is determined by the first portion of the hashed query fragment. The two OID fields contain an object type specifier and an 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 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.

Considering next the Communication Module contained in the computer nodes used in the preferred embodiment, refer to Fig. 5, 6 and 7. 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 and objects to the home node, and the home node transmits results, such as formatted tables, to the Front End node.

Considering next the modules contained in the home nodes used in the preferred embodiment, refer to Fig. 5. The Query Parser parses a query into a query computation tree. The nodes of the query computation tree are either internal nodes or leaf nodes. An internal node is a node having one or more child nodes. An internal node specifies how the results of the child nodes are to be combined. A leaf node is a node having no children. A leaf node is either a constant value or an elementary query. Elementary queries are of three kinds: index queries, link queries and object queries. Each elementary query consists of a command that determines what action the elementary query is to perform and additional information that depends on the kind of elementary query. An object query contains an OID, and the other two kinds of elementary query contain an information object conforming to the ontology of the object database. A link query contains exactly one link to an unspecified information object. An index query does not contain any unspecified links to other information objects. The query computation tree is transferred to the Query Processor.

The Query Processor is responsible for administering the processing of the query. Upon receiving a query computation tree from the Query Parser, it assigns a query identifier (QID) to the query, and it assigns an elementary query identifier (EQID) to each leaf node that specifies an elementary query. Each elementary object query

is transmitted to an object node using an Object Query Message. The elementary index and link queries are transferred to the Feature Extractor. As Query Response Messages are received, the processing specified in the query computation tree is performed. When the entire query has been computed, the response is formatted and transmitted to the front end from which the query was received.

The Feature Extractor extracts features from an object or elementary query. 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 extracted features 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 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. If the fragment is derived from an elementary index query or an elementary link query, the Hashing Module transmits an Index Query Message or Link Query Message to the Communication Module, respectively. If the fragment is derived from an object, then the Hashing Module transmits an Insert Link Message when the fragment includes a link and the Hashing Module transmits an Insert Index Message when the fragment does not include a link.

The Similarity Comparator gathers all the query responses for each elementary query. For each object or link in the responses, the Similarity Comparator determines the relevance of each object or link, respectively, returned in the search. This determination of relevance is made by the home node by comparing the degree of similarity between the elementary query and the objects or links, respectively, whose OIDs or pairs of OIDs, respectively, were received. 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 or pairs of OIDs, respectively, are transferred to the Query Processor.

Considering next the modules contained in the index nodes used in the preferred embodiment, refer to Fig. 6. The Fragment Table receives Index Query Messages, Index Link Messages, Insert Index Messages and Insert Link Messages. In the case of

an Insert Query Message or an Insert Link 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. In the case of an Insert Index Message or an Insert Link Message, the Fragment Table modifies an entry in the local hash table by adding the OID or pair of OID fields, respectively, and the Value field of the Insert Index Message or the Insert Link Message, respectively, to the entry in the local hash table.

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 an Index Query Response Message or a Link Query Response Message, depending on whether the entry received from the Fragment Table is an index entry or link entry, respectively.

Considering next the module contained in the object nodes used in the preferred embodiment, refer to Fig. 7. The Object Table receives Object Query Messages and Insert Object Messages. In the case of an Object Query Message, the Object Table retrieves an entry in the local table using the object identifier in the OID field of the Object Query Message. The Object Query Message and the retrieved entry are transmitted to the Communication Module using an Object Query 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 a query language in a distributed computer database system having a plurality of home nodes and a plurality of index nodes connected by a network, said method comprising the steps of:

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

(b) parsing, by said selected home node, a query conforming to the said query language, from a user, to obtain a plurality of elementary queries;

(c) each of said elementary queries being an index query or a link query;

(d) extracting, by said selected home node, a plurality of features from each elementary query of the said plurality of elementary queries;

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

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

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

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

(i) returning, by each said index node accessing data according to said respective hashed link query feature a plurality of pairs 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 parsing 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 each said elementary query; and

(b) returning to said user or using for subsequent processing, by said home node, accessed data having a degree of similarity determined by the said elementary query.

subsequent to the step of returning said plurality of object identifiers or said plurality of pairs of object identifiers, according to said respective hashed index feature or hashed link feature.

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 elementary query; and

(b) features possessed only by the said elementary query.

5. A method of storing objects or locations of objects in a manner which is conducive to information retrieval using a query language in a distributed computer database system having a plurality of home nodes and a plurality of index 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) each of said plurality of features is either an index feature or a link feature;

(d) 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;

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

(f) using, by said index node, said second portion of said respective hashed object feature to store data according to a local hash table located on said index node, said data consisting of the object identifier of the object containing the feature and, in addition, if the feature is a link feature, the object identifier of the object referenced by the link contained in the link feature.

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 index nodes;

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

(d) wherein each said home node, upon receiving a query from a user, parses the said query to obtain a plurality of elementary queries, each of which is either an index query or a link query, extracts a plurality of features from each said elementary query, hashes each said elementary query feature of said plurality of query features into a hashed elementary query feature having a first portion and a second portion, and transmits each said hashed query feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed elementary query feature,

(e) further wherein each said index node uses said second portion of said hashed query feature to access data according to a local hash table located on said index node and returns a plurality of object identifiers or a plurality of pairs 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 elementary query and returns to said user, or uses for subsequent processing, 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 elementary query; and

(b) features possessed only by the said elementary 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 index nodes;

(c) said plurality of home nodes and said plurality of index 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, each feature of said plurality of object features being either an index feature or a link feature, 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 index nodes indicated by said first portion of said hashed object feature,

(e) further wherein each said index node uses said second portion of said hashed object feature to store object identifiers according to a local hash table located on said index 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 index nodes;

(c) said plurality of home nodes and said plurality of index 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, parsing the said query into a plurality of elementary queries, each of said elementary queries being either an index query or a link query, extracting a plurality of features from each said elementary query of said plurality of elementary queries parsed from the said query contained in said query command, hashing each said elementary query feature of said plurality of elementary query features into a hashed elementary query feature having a first portion and a second portion, and transmitting an elementary query message containing each said hashed elementary query feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed elementary query feature,

(f) said index node, upon receipt of said elementary query message, using said second portion of said hashed elementary query feature to access data according to a local hash table located on said index node and transmitting a message returning a plurality of object identifiers, if the said elementary query feature is an index feature, or returning a plurality of pairs of object identifiers, if the said elementary query feature is a link feature, corresponding to said accessed data to said home node.

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

(a) a plurality of home nodes; and

(b) a plurality of index nodes;

(c) said plurality of home nodes and said plurality of index 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, each said feature of said plurality of features being either an index feature or a link feature, 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 index nodes indicated by said first portion of said hashed object feature,

(f) said index 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 index node, said data consisting of the object identifier of the object containing the feature and, in addition, if the feature is a link feature, the object identifier of the object referenced by the link contained in the link feature.

13. A method for information retrieval using a query language in a distributed computer database system having a plurality of home nodes, a plurality of index 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) parsing, by said selected home node, a query conforming to the said query language, from a user, to obtain a plurality of elementary queries;

(c) each of said elementary queries being either an index query or an object query;

(d) extracting, by said selected home node, a plurality of features from each index query of the said plurality of elementary queries;

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

(f) extracting, by said selected home node, an object identifier from each object query of the plurality of elementary queries, said object identifier having a first portion and a second portion;

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

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

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

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

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

(l) returning, by each said object node accessing data according to the said object identifier contained in said respective object query, a plurality of object identifiers corresponding to said accessed data 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 parsing 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 index query; and

(b) returning to said user or using for subsequent processing, by said home node, accessed data having a degree of similarity determined by the said index query,

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

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 index query; and

(b) features possessed only by the said index query.

17. A method of storing objects or locations of objects in a manner which is conducive to information retrieval using a query language in a distributed computer

database system having a plurality of home nodes, a plurality of index 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 from an object submitted by a user;

(c) extracting, by said selected home node, a plurality of object identifiers of the objects referenced by the said object submitted by a user, each said object identifier having a first portion and a second portion;

(d) 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;

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

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

(g) transmitting, by said selected home node, the object identifier of the said object submitted by a user, to the plurality of object nodes indicated by said first portion of each said object identifier of the said plurality of object identifiers of the objects referenced by the said object submitted by a user;

(h) using, by said object node, said second portion of each said object identifier of the said plurality of object identifiers to store data according to a local table located on said object node;

(i) transmitting, by said selected home node, the said plurality of object identifiers of the objects referenced by the said object submitted by a user to a respective one of said plurality of object nodes indicated by said first portion of said object identifier of the said object submitted by a user; and

(j) using, by said object node, said second portion of the said object identifier of the said object submitted by a user, to store data according to a local table located on said object node, said data consisting of the plurality of object identifiers of the objects referenced by the said object submitted by a user.

18. The method of claim **17** 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.

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 index nodes; and

    (c) a plurality of object nodes;

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

    (e) wherein each said home node, upon receiving a query from a user, parses the said query to obtain a plurality of elementary queries, each of which is either an index query or an object query, extracts an object identifier from each said object query, each said object identifier having a first portion and a second portion, extracts a plurality of features from each said index query, hashes each said index query feature of said plurality of index query features into a hashed index query feature having a first portion and a second portion, transmits each said hashed index query feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed query feature, 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,

    (f) further wherein each said index node uses said second portion of said hashed query feature to access data according to a local hash table located on said index node and returns a plurality of object identifiers corresponding to said accessed data to said home node, each said object node uses said second portion of said object identifier to access data according to a local table located on said object node and returns a plurality of object identifiers corresponding to 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 elementary query and returns to said user, or uses for subsequent processing, 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 index query; and

    (b) features possessed only by the said index query.

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 index nodes; and

    (c) a plurality of object nodes;

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

    (e) 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, extracts a plurality of object identifiers of the objects referenced by the said object, each said object identifier having a first portion and a second portion, transmits each said hashed object feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed object feature, transmits the object identifier of the said object to the plurality of object nodes indicated by said first portion of each said object identifier of the said plurality of object identifiers, transmits the said plurality of object identifiers to a respective one of said plurality of object nodes indicated by the said first portion of the said object identifier,

    (f) further wherein each said index node uses said second portion of said hashed object feature to store objects or locations of objects according to a local hash table located on said index node, each said object node uses said second portion of said object identifier to store data according to a local table located on said object 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 index nodes; and

    (c) a plurality of object nodes;

    (d) said plurality of home nodes, said plurality of index 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, parsing the said query into a plurality of elementary queries, each of said elementary queries being either an index query or an object query, extracting a plurality of features from each said index query of said plurality of elementary queries parsed from the said query contained in said query command, extracting an object identifiers from each said object query of said plurality of elementary queries parsed from the said query contained in said query command, each said object identifier having a first portion and a second portion, hashing each said index query feature of said plurality of index query features into a hashed index query feature having a first portion and a second portion, transmitting an index query message containing each said hashed index query feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed index query feature, transmitting an object query message containing said object identifier to a respective one of said plurality of object nodes indicated by said first portion of said object identifier,

(g) said index node, upon receipt of said index query message, using said second portion of said hashed index query feature to access data according to a local hash table located on said index node and transmitting a message returning a plurality of object identifiers corresponding to said accessed data to said home node, said object node, upon receipt of said object query message, using said second portion of said object identifier to access data according to a local table located on said object node and transmitting a message returning a plurality of object identifiers corresponding to said accessed data to said home node.

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

(a) a plurality of home nodes;

(b) a plurality of index nodes; and

(c) a plurality of object nodes;

(d) said plurality of home nodes, said plurality of index 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) 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, extracting a plurality of object identifiers of the objects referenced by the said object, each said object identifier having a first portion and a second portion, 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 insert message containing each said hashed object feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed object feature, transmitting an insert message containing the said object identifier to the plurality of object nodes indicated by said first portion of each said object identifier of the said plurality of object identifiers, transmitting an insert message containing the said plurality of object identifiers to a respective one of said plurality of object nodes indicated by said first portion of said object identifier,

(g) said index node, upon receipt of said insert message, using said second portion of said hashed object feature or of said object identifier to store data according to a local hash table or local table located on said index node or object node.

25. The method of claim **1** combined with the method of claim **13** wherein an elementary query may be an index query, a link query or an object query.

26. The method of claim **25** further comprising the step of receiving, at said home node, said query from said user, prior to the step of parsing said query.

27. The method of claim **26** further comprising the steps of:

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

(b) returning to said user or using for subsequent processing, by said home node, accessed data having a degree of similarity determined by the said index query,

subsequent to the step of returning said plurality of object identifiers or said plurality of pairs of object identifiers, according to said respective hashed index feature or hashed link feature.

28. The method of claim **27** 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 index query; and

(b) features possessed only by the said index query.

29. The method of claim **5** combined with the method of claim **17** wherein an object feature may be either an index feature or a link feature, and data is stored on both index nodes and object nodes.

30. The method of claim **29** 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.

31. The method of claim **7** combined with the method of claim **19** wherein an elementary query may be an index query, a link query or an object query.

32. The distributed computer database system of claim **31** wherein said home node determines a measure of similarity between said accessed data and said elementary query and returns to said user, or uses for subsequent processing, accessed data having a predetermined degree of similarity.

33. The method of claim **32** wherein said home node measures similarity using a similarity function determined by:

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

    (b) features possessed only by the said index or link query.

34. The method of claim **10** combined with the method of claim **22** wherein an object feature may be either an index feature or a link feature, and data is stored on both index nodes and object nodes.

35. The method of claim **11** combined with the method of claim **23** wherein an elementary query may be an index query, a link query or an object query.

36. The method of claim **12** combined with the method of claim **24** wherein an object feature may be either an index feature or a link feature, and data is stored on both index nodes and object nodes.

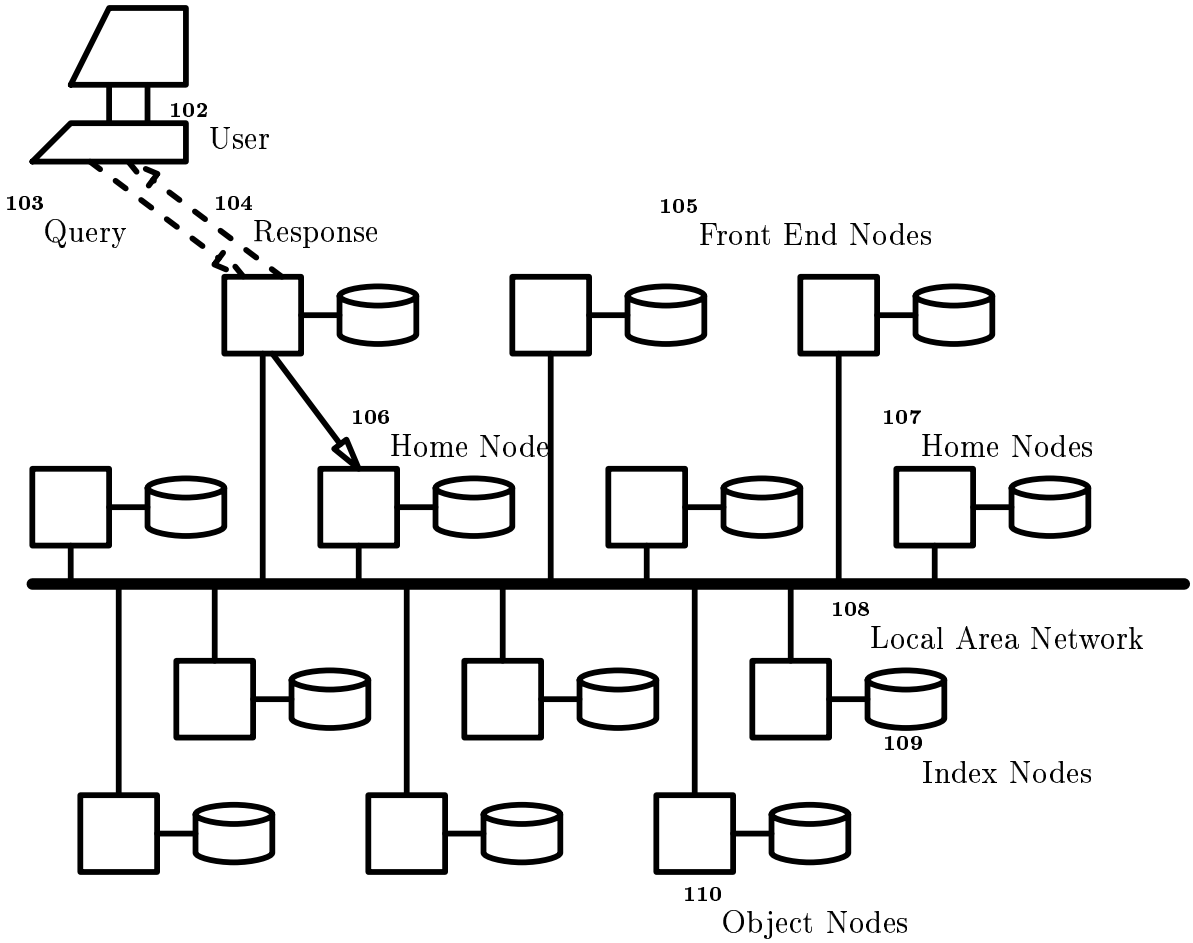
102
User
103
Query
104
Response
105
Front End Nodes
106
Home Node
107
Home Nodes
108
Local Area Network
109
Index Nodes
110
Object Nodes

FIG. **1**

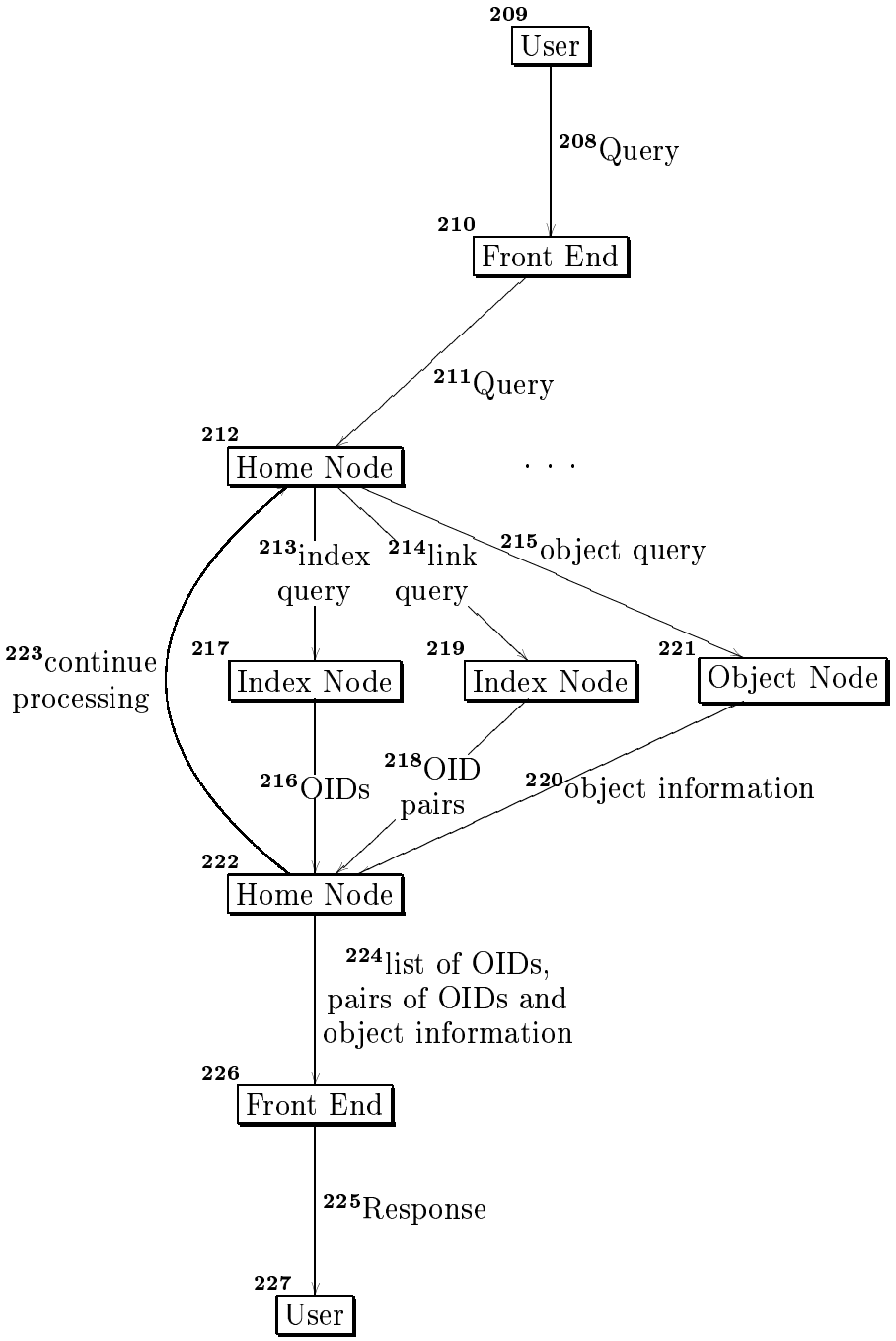
209
User
208 Query
(Step 201)
210
Front End
211 Query
(Step 202)
212
Home Node
. . .
213 index query
214 link query
215 object query
(Step 203)
223 continue processing
217
Index Node
219
Index Node
221
Object Node
216 OIDs
218 OID pairs
220 object information
(Step 204)
222
Home Node
224 list of OIDs, pairs of OIDs and object information
(Step 205)
226
Front End
225 Response
(Step 206)
227
User

FIG. **2**

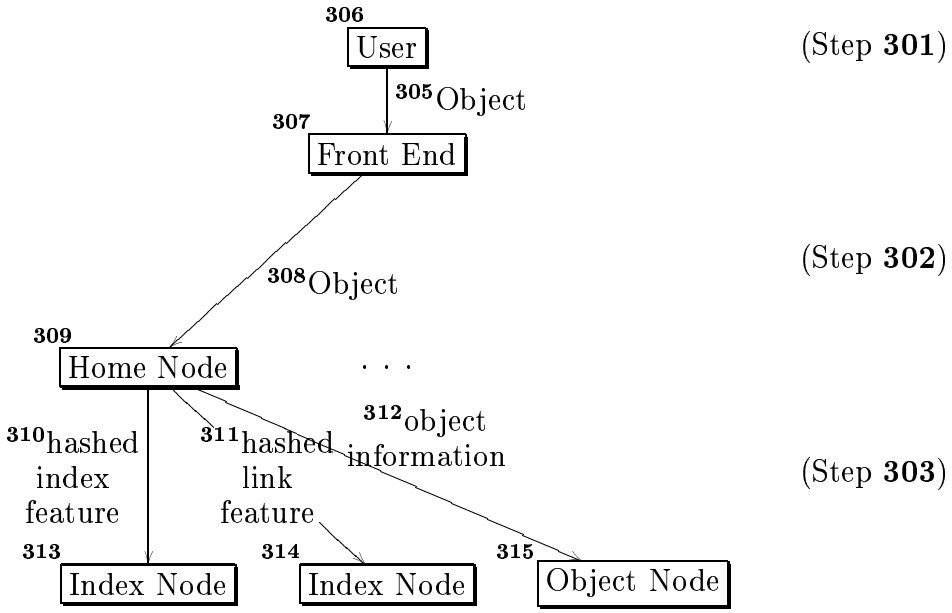

FIG. **3**

Index Query Message

402 Header	403 EQID	404 HQF	405 Value

FIG. **4a**

Index Query Response Message

406 Header	407 EQID	408 OID	409 Weight

FIG. **4b**

Link Query Message

410 Header	411 EQID	412 HQF	413 Value

FIG. **4c**

Link Query Response Message

414 Header	415 EQID	416 OID1	417 OID2	418 Weight

FIG. **4d**

Object Query Message

419 Header	420 EQID	421 OID

FIG. **4e**

Object Query Response Message

422 Header	423 EQID	424 OID	425 Location
426 Features...			
427 ...			

FIG. **4f**

Insert Index Message

428 Header	429 OID	430 HQF	431 Value

FIG. **4g**

Insert Link Message

432 Header	433 OID1	434 OID2	435 HQF	436 Value

FIG. **4h**

Insert Object Message

437 Header	438 OID	439 Location
440 Features...		
441 ...		

FIG. **4i**