Distributed Object Search System and Method

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 including images, sound and video streams, as well as plain and structured text. A query is an object in the same format as the objects to be retrieved. 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 and hashes these features. Each hashed feature 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 are gathered by the home node. If requested, this process is repeated a second time 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] A. Del Bimbo, editor. *The Ninth International Conference on Image Analysis and Processing*, volume 1311. Springer, September 1997.

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

[5] P. Hayes and J. Carbonell. Scout - automated query-relevant document summarization. Technical Report 1997 Project Summary, Carnegie Group, Pittsburgh, PA, 1997.

[6] 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.

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

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

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

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

[11] M. Zloof. Query-by-example: the invocation and definition of tables and forms. In *Proc. Conf. on Very Large Databases*, pages 1–24, 1975.

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

Two of the most significant changes in the nature of information processing in the last decade are the transition from primarily alphanumeric text processing to multimedia processing and the connection of formerly isolated computers by networks, which have been connected in turn by intranets and the Internet. The first change has resulted in computer images becoming as common on computers as text. The second change has resulted in vast quantities of information, both text and multimedia, being accessible to individuals.

The second change has increased the resources available to an individual but at the cost of the increasing difficulty of finding relevant information. Search engines have been developed to assist in this task, but they are still primarily based on matching words in a query with words in text documents. In particular, this means that current search engines cannot effectively search for features of images and other kinds of multimedia.

For example, in the field of medicine, lung cancer is one of the most difficult cancers to cure. Early detection is important to improve the recovery rate. Chest CT scans are more effective than conventional chest X-ray images, but CT images

result in many more images to be examined, making computer assistance essential for mass screening programs. Computer aided diagnosis of CT images requires the extraction of a large number of features such as the lung area, blood vessels, air clusters and tumors. These features are detected using a thresholding algorithm along with smoothing to remove artifacts of the CT scanner. These features, in turn, have a complex structure involving attributes such as their shape, area, thickness and position within the lung.

An object database consists of a collection of data or information objects. 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.

Databases containing images, sound and/or video streams, include not only the information objects themselves but also features and metadata. The data model used for such a database supports the representation of information at many levels of abstraction, including:

1. The *data representation level* contains the actual data of the information object.

2. The *data object level* stores data objects (such as lines and regions) extracted from the information object. The objects on this level do not have a domain interpretation.

3. The *domain object level* associates a domain object with each object at the data object level.

4. The *domain event level* associates domain objects with each other, providing the semantic representation of spatial or temporal relationships.

In general, a *feature* is any information or knowledge associated with an information object or derived from the content of the information object. Each feature is represented in a computer system using a data structure. A feature at the data object level consists of domain-independent data such as lines and regions (i.e., at Level 2 above). A feature at the domain level consists of domain objects related to one another by domain-dependent relationships (i.e., Levels 3 and 4 above).

In medicine, mammography is one of the most effective methods of early detection of breast cancer, one of the leading causes of cancer among women. Manual reading of mammograms is labor intensive, so computer assistance is essential. A very large number of features have been identified as being important for proper diagnosis, such as clustered microcalcifications, stellate lesions and tumors. Each of these is a medical domain object with a complex structure. For example, a stellate lesion has a complex structure, consisting of a central mass surrounded by spicules. The spicules, in turn,

have a complex, star-shaped stucture. Extracting these complex domain objects and their relationships with each other is important for effective detection of breast cancer.

The features that are stored in a database containing information objects such as images, sound and video streams are of the following types:

- Features, such as the name of the photographer or date taken, that cannot be directly extracted from an information object. Features of this kind are called *metadata*.
- Features extracted directly from the information object at the time of insertion into the database.
- Features that are not calculated until needed.

Features can be as simple as the value of an attribute such as the brightness of an image, but many features must be represented using a complex data structure such as a representation of the structure of a stellate lesion in a mammogram.

Features are extracted from structured documents by parsing the document to produce data structures. Features are extracted from unstructured documents by using one of the many feature extraction algorithms that have been developed. As in the case of structured documents, feature extraction from an unstructured document produces data structures.

A large variety of feature extraction algorithms have been developed for media such as sound, images and video streams. For example, medical images typically use edge detection algorithms to extract the data objects, while domain-specific knowledge is used to classify the data objects as medically significant objects, such as blood vessels, lesions and tumors. Fourier and Wavelet transformations as well as many filtering algorithms are also used for feature extraction. For example, wavelet analysis has been used to characterize the texture of a region and to determine a shape (such as a letter) no matter where or in what orientation the shape has within the image.

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.

The data structures that represent the features conform to a *data model* that determines the kinds of components and attribute values that are allowed. The data model for features at the domain level is often called an *ontology*. 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.

Current systems that perform feature extraction from information objects use very simple ontologies. Furthermore the ontologies are implicit in the design of the system rather than being a separate component of the system. As a result, current systems cannot be used for more than the single ontology for which it was designed. Using a different ontology or even adding new capabilities to the ontology are generally not possible without completely redesigning the system. Such systems are not suited for the large, complex, evolving ontologies that are typical of modern application domains.

A variety of similarity measures have been developed for comparing a query object with an object in the database. When the query and information object are documents in a natural language, the most commonly used measure of similarity is the cosine measure. The cosine measure is given by the formula $\cos(v, w)$, where the vector v denotes the query and the vector w denotes the information object. These vectors are in a space in which each possible word (or set of synonymous words) represents one dimension of the space.

When the information object is not a document written in a natural language, the cosine measure is not applicable and other measures have been developed. Most of the measures that have been developed are distance functions. However, there is convincing evidence that human perception of similarity does not satisfy the axioms of a distance function. The model that is currently 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.

The Feature Contrast Model is designed for only the simplest kind of feature, namely a feature that an information object either possesses or does not possess. To extend the Feature Contrast Model to more general features, it is necessary to define functions that compare features to determine whether a match occurs at all (feature matching functions) and the strength of a match when it has been determined that a match exists.

To assist in finding information in a database, special search structures are employed called *indexes*. Indexing techniques as currently developed are very limited when it comes to addressing the problem of similarity indexing. Many search engines are limited to indexing the metadata attached to the information objects and do not index the content of the information objects. Search engines that directly index the content of information objects use indexing techniques that degrade significantly with increase in dimension, and they generally just select some of the information objects rather than rank order them.

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.

Another characteristic of current technology that it treats each information object as an indivisible unit for the purposes of retrieval: an information object is retrieved as a whole or not at all. For example, World Wide Web browsers retrieve each document as a unit and present it only when the entire document has been downloaded and formatted. Individual data entries and even sections within an object are not individually indexed. Some search engines are even more extreme in this respect. They only categorize entire Web sites.

Current search engines commonly include index entries that are *stale*, i.e., the documents that produced the index entries have been updated or deleted since the time when the documents were indexed. This behavior is necessary because it is prohibitively expensive to monitor so many documents continuously. In most cases, this behavior is acceptable, but for certain time-sensitive documents, such as those containing commodity prices, it is important for the index to be current.

3 Summary of the Invention

The present invention supports indexing and retrieval of information objects which can be images, sound and video streams, as well as objects such as text files and structured documents. Furthermore, the present invention can index both the content of the information objects themselves as well as any metadata attached to the objects.

In the present invention, the retrieval of objects relevant to a query is based on an ontology, which is a separate component of the system. The present invention can support ontologies which are arbitrarily large, complex and evolving.

In the present invention, 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 URL. One can indicate that an information object is time-sensitive, in which case the object will be downloaded and processed when it is relevant to a query.

In the present invention queries to retrieve objects in the database are in the same format as the objects in the database, except that a query may include one or more *markers* denoting data items to be retrieved. Such markers are used in relational forms languages such as Query-by-Example. The present invention extends this technique to fuzzy queries on object databases. The features of the queries are extracted using the same ontology as the one used to extract features from the information objects.

The present invention supports the indexing of all three kinds of features: meta-data, features computed when the object is indexed and features computed during query processing. The features indexed by the present invention may be arbitrarily complex data structures. Virtually any similarity measure may be used to compare a query with an information object, and more than one similarity measure may be used within the same query or information object. In particular, the Feature Contrast Model, extended to the case of any kind of feature, is supported by using functions associated with the types of feature allowed by the ontology. The present invention uses the object-oriented software paradigm to associate functions with feature types. The association of functions to types is one of the basic properties of object-oriented systems.

The present invention uses a high-performance distributed indexing methodology that scales up very well to support the indexing of very large numbers of 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 to the system. This avoids the limitations of current systems.

The present invention supports indexing and retrieval of individual data entries within a single information object. Data entries from a collection of relevant sources can be collected into a single table for presentation to the user. Furthermore, the user may specify that the requested information is time-sensitive, in which case the present invention will download the current state of the information object and process it to extract the relevant information. This avoids the limitation of current search engines that contain large numbers of stale index entries.

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. The node receiving the query, termed the home node of the search engine, extracts the features of the received query using the feature extraction algorithms specified in the ontology. The features are fragmented into data structures having a bounded size. The fragments are then hashed using one of the many hashing algorithms that are available. A portion of each hashed fragment 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.

Each node on the network which receives a hashed query fragment uses the hashed query fragment to perform a search on its respective database. Nodes finding data corresponding to the hashed query fragment return the OIDs of the objects possessing this fragment. A matching function specific to the type of the fragment may be invoked to select a subset of the OIDs to be returned.

The home node gathers the extracted information and a similarity function is computed based on the fragments that are in common with the query as well as the fragments that are in the query but not in the object. The similarity function is used to rank the objects, and it can be based on functions which can be used to compute the strength of the match. The function used for each fragment is specific to the type of the fragment. The results are either a list of object identifiers in rank order or a table of data associated with or extracted from the objects.

The home node can also reduce redundancy when the same information is contained in more than one document. In particular, the extracts can be arranged according to the Maximum Marginal Relevance (MMR) metric of Hayes and Carbonell.

The results, whether a list or a table, are transmitted to the front end node which formats the response to the user. 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 list of URLs or a table each of whose entries have the extracted parts of a relevant document as well as a reference to the URL of the document. The front end transmits the formatted response to the user.

Information objects to be indexed are processed in the same manner as queries, except that the query nodes simply store data in their respective databases and no information is returned to the home 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 additional 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 fragments 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 value can use functions specific to the types of the fragments. 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.

Level 3 service has the additional capability of downloading and processing the original information object if this is specified. There are three ways that it can be specified:

1. The ontology can specify that a type of fragment is time-sensitive.

2. The information object itself can specify that it is time-sensitive.

3. The query can specify that some or all of its fragments are time-sensitive.

In each case above, the information object must be downloaded if it is requested and the most recent download is older than a specified amount.

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 of 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 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 through a network from disk server or other external servers.

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 preferred embodiment.

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. 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 queries or documents by parsing the query or document to produce a data structure, then dividing this data structure into (possibly overlapping) substructures called fragments. Fragments of a query are used to find matching fragments in the database, so they are also called *probes*.

Features are extracted from unstructured 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.

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.

When a marker appears in a query fragment, the marker represents a data item to be retrieved. The marker is replaced by any term in a document such that the modified fragment is a valid fragment according to the ontology. For example, a fragment that

requests the color of a house would retrieve all colors of houses occurring in the object database.

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 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 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 data is to be sent to be stored or to which a query fragment 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 fragments 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 fragments by which the data was initially stored on that query node respond to the 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 hashed the query fragments.

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. 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.

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 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 and then returns a list of the most relevant OIDs. In one embodiment the list 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 list of the most relevant OIDs is transmitted directly to the user computer through 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. According to a time sensitivity specification, an object node may

download the object from an external server to update the object information maintained on the object node. Downloading is accomplished by establishing communication with the external server responsible for the object, retrieving the object, and extracting the features of the object (Step **206**). The time sensitivity specification may be specified in the query, in each fragment of the query and in the object.

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 through the network without the intervention of the front end computer.

For level **3** service, the object nodes transmit the object information of the 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 and may also result in a different list of objects being returned. 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 must contain 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 (Step **209**). In another embodiment the home node uses uses the extracted information of the relevant objects to construct one or more tables of information. In another embodiment the object information of the relevant objects or the constructed tables are transmitted directly to the user computer through 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 next the message formats used in the preferred embodiment, refer to FIG. **4**. The Query Message is produced by the Hashing Module and transmitted from a home node to a query node. The Query Message 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 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 Query Response Message is produced by the Similarity Comparator and transmitted from a query node to a home node. Each Query Response Message is the result of a Query Message. 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 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 is produced by the Similarity Comparator and transmitted from a home node to an object node. The Object Message has four fields: Header, QID, OID and Time Sensitivity (TS). 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 TS field contains an optional time sensitivity specifier. The object type specifier determines whether the Object Message contains a TS field, and if the Object Message does contain a TS field then the object type specifier determines the size of the TS field.

The Object Response Message is produced by the Object Table or by the Feature Extractor and transmitted from an object node to a home node. 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 is produced by the Hashing Module and transmitted from a home node to a query node. The Insert Message 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 query 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 Object Message is produced by the Feature Extractor and transmitted from a home node to an object node. The Insert Object Message has three parts: Identifier, Feature and Auxiliary. The Identifier part has four fields: Header, OID, TS 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 TS field contains an optional time sensitivity specifier. The object type specifier determines whether the Object Message contains a TS field, and if the Object Message does contain a TS field then the object type specifier determines the size of the TS field. 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 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 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. 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 Table Constructor receives Object Response Messages. It formats a table 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.

Considering next the modules contained in the query nodes used in the preferred embodiment, refer to Fig. 6. 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. 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.

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.

Considering next the modules contained in the object nodes used in the preferred embodiment, refer to Fig. 7. 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 transferred to the Download Determiner. 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.

The Download Determiner receives Object Messages and entries from the Object Table. It uses the TS field to determine whether the object should be downloaded using the Downloader. In one embodiment, the TS field is one bit whose value has

two states corresponding to the two possibilities of downloading or not downloading. In another embodiment, the TS field is an expiration time. When the expiration time has been reached, the object is downloaded. Otherwise the object is not downloaded. If the Download Determiner determines that the object should not be downloaded, then the Object Message and entry received from the Object Table is transmitted to the Communication Module using an Object Response Message. If the Download Determiner determines that the object should be downloaded, then the Object Message received from the Object Table is transferred to the Downloader.

The Downloader receives an Object Message from the Download Determiner. The Downloader uses the Location field of the Object Message to download the object. In one embodiment, the Downloader uses the Hypertext Transfer Protocol (HTTP) to download a Web page specified by a Universal Resource Locator (URL). The downloaded object is transmitted to the Feature Extractor.

The Feature Extractor extracts features from an object received from the Downloader. 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 Object Table using an Insert Object Message which replaces causes an entry in the local object table to be replaced. The extracted features are also transmitted to the Communication Module using an Object Response 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 from a query by a user;

   (c) fragmenting, by said selected home node, each said extracted feature of said plurality of extracted features into a plurality of query fragments;

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

(e) transmitting, by said selected home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment;

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

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

2. [OO] The method of claim **1** further comprising the step of applying a matching function to the said accessed data to select a portion of the said plurality of object identifiers, said matching function being specific to the type of the said query fragment, prior to the step of returning the said portion of the said plurality of object identifiers to said selected home node.

3. 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.

4. The method of claim **3** 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.

5. The method of claim **4** 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.

6. [OO] The method of claim **5** wherein for each feature of said plurality of features, said similarity function uses a function specific to the type of the said feature.

7. 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) fragmenting, by said selected home node, each said extracted feature of said plurality of extracted features into a plurality of object fragments;

   (d) hashing, by said selected home node, each said object fragment of said plurality of object fragments, said hashed object fragment having a first portion and a second portion;

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

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

8. The method of claim **7** 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.

9. 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 from said query, fragments each said query feature of said plurality of query features into a plurality of query fragments, hashes each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment, and

(e) further wherein each said query node uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node and returns a plurality of object identifiers corresponding to said accessed data to said home node.

10. [OO] The method of claim **9** wherein said query node applies a matching function to the said accessed data to select a portion of the said plurality of object identifiers, said matching function being specific to the type of the said query fragment.

11. The distributed computer database system of claim **9** 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.

12. The method of claim **11** 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.

13. [OO] The method of claim **12** wherein for each feature of said plurality of features, said similarity function uses a function specific to the type of the said feature.

14. 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, fragments each said object feature of said plurality of object features into a plurality of object fragments, hashes each said object fragment of said plurality of object fragments into a hashed object fragment having a first portion and a second portion, and transmits each said hashed object fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed object fragment, and

(e) wherein each said query node uses said second portion of said hashed object fragment to store objects or locations of objects according to a local hash table located on said query node.

15. [Version A] 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 from a query contained in said query command, fragmenting each said query feature of said plurality of query features into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmitting a query message containing each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

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

16. [Version B] A computer database system having an information retrieval tool for handling queries from a user.

(a) Said computer database system, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

(b) a query task enqueued being resultant in, in response to a query command from said user, extracting a plurality of features from a query contained in said query command, fragmenting each said query feature of said plurality of query features into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments, and transmitting an access command for each hashed query fragment.

(c) Said computer database system, upon receipt of said access command, enqueuing a predetermined access task in response to said access command,

(d) an access task enqueued being resultant in, in response to an access command, using said hashed query fragment to access data according to a hash table and returning a plurality of object identifiers corresponding to said accessed data.

17. [Version A] 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, fragmenting each said object feature of said plurality of object features into a plurality of object fragments, hashing each said object fragment of said plurality of object fragments into a hashed object fragment having a first portion and a second portion, and transmitting an insert message containing each said hashed object fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed object fragment,

(f) said query node, upon receipt of said insert message, using said second portion of said hashed object fragment to store data according to a local hash table located on said query node.

18. [Version B] A computer database system for storage and retrieval of information.

(a) Said computer database system, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

(b) 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, fragmenting each said object feature of said plurality of object features into a plurality of object fragments, hashing each said object fragment of said plurality of object fragments, and transmitting an insert command for each hashed object fragment.

(c) Said computer database system, upon receipt of said insert command, enqueuing a predetermined insert task in response to said insert command,

(d) an insert task enqueued being resultant in, in response to an insert command, using said hashed object fragment to store data according to a hash table.

19. 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 from a query by a user;

(c) fragmenting, by said selected home node, each said extracted feature of said plurality of extracted features into a plurality of query fragments;

(d) hashing, by said selected home node, each said query fragment of said plurality of query fragments, said hashed query fragment having a first portion and a second portion;

(e) transmitting, by said selected home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment;

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

(g) returning, by each said query node accessing data according to said respective hashed query fragment, 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.

20. [OO] The method of claim **19** further comprising the step of applying a matching function to the said accessed data to select a portion of the said plurality of object identifiers, said matching function being specific to the type of the said query fragment, prior to the step of returning the said portion of said plurality of object identifiers to said selected home node.

21. [Download] The method of claim **19** further comprising the steps of

    (a) downloading, by said object node, the object determined by said respective object identifier, from an external server located by said accessed data, if current object information is not already included in said accessed data;

    (b) extracting, by said object node, data from said object according to said query;

    prior to the step of returning an object location, object features and other auxiliary information to said selected home node.

22. The method of claim **19** 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.

23. [Time Sensitivity] The method of claim **22** wherein said query from said user contains a time sensitivity requirement specification.

24. The method of claim **22** 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.

25. [Table] The method of claim **24** further comprising the step of constructing, by said selected home node, a table containing, for each object of the plurality of objects, the said object location, said plurality of object features and said auxiliary information, subsequent to the step of returning accessed data to said selected home node.

26. The method of claim **24** 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.

27. [OO] The method of claim **26** wherein for each feature of said plurality of features, said similarity function uses a function specific to the type of the said feature.

28. 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) fragmenting, by said selected home node, each said extracted feature of said plurality of extracted features into a plurality of object fragments;

(f) hashing, by said selected home node, each said object fragment of said plurality of object fragments, said hashed object fragment having a first portion and a second portion;

(g) transmitting, by said selected home node, the location of the said object, the said plurality of 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 each said object identifier;

(h) 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;

(i) transmitting, by said selected home node, each said hashed object fragment of said plurality of fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed object fragment; and

(j) using, by said query node, said second portion of said respective hashed object fragment to store data according to a local hash table located on said query node.

29. [Time Sensitivity] The method of claim **28** wherein said object contains a time sensitivity requirement specification.

30. The method of claim **28** 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. [Time Sensitivity] The method of claim **30** wherein said user specifies a time sensitivity requirement specification for said object.

32. 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 from said query, fragments each said query feature of said plurality of query features into a plurality of query fragments, hashes each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

    (f) further wherein each said query node uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node and returns a plurality of object identifiers corresponding to said accessed data to said home node,

    (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.

33. [OO] The method of claim **32** wherein said query node applies a matching function to the said accessed data to select a portion of the said plurality of

object identifiers, said matching function being specific to the type of the said query fragment.

34. [Download] The method of claim **32** wherein said object node downloads the object determined by said respective object identifier, from an external server located by said accessed data, if current object information is not already included in said accessed data, and extracts data from said object according to said query.

35. The distributed computer database system of claim **32** 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.

36. [Time Sensitivity] The method of claim **35** wherein said query from said user contains a time sensitivity requirement specification.

37. [Table] The method of claim **35** wherein said selected home node constructs a table containing, for each object of the plurality of objects, the said object location, said plurality of object features and said auxiliary information.

38. The method of claim **35** 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.

39. [OO] The method of claim **38** wherein for each feature of said plurality of features, said similarity function uses a function specific to the type of the said feature.

40. 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, fragments each said object feature of said plurality of object features into a plurality of object fragments, hashes each said object fragment of said plurality of object fragments into a hashed object fragment having a first portion and a second portion, transmits the location of the said object, the said plurality of 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 fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed object fragment,

(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 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 fragment to store objects or locations of objects according to a local hash table located on said query node.

41. [Time Sensitivity] The method of claim **40** wherein said object contains a time sensitivity requirement specification.

42. [Version A] 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 from a query contained in said query command, fragmenting each said query feature of said plurality of query features into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments into a hashed query fragment having a first portion and a second portion, and transmitting a

query message containing each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment,

(g) said query node, upon receipt of said query message, using said second portion of said hashed query fragment to access data, consisting of 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 and transmitting a message returning the said plurality of object identifiers corresponding to said accessed data to said home node,

(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 object features of said said object and auxiliary information associated with the said object, corresponding to said accessed data to said home node.

43. [Version B] A computer database system having an information retrieval tool for handling queries from a user.

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

(b) a query task enqueued being resultant in, in response to a query command from said user, extracting a plurality of features from a query contained in said query command, fragmenting each said query feature of said plurality of query features into a plurality of query fragments, hashing each said query fragment of said plurality of query fragments, and transmitting an access command for each hashed query fragment.

(c) Said computer database system, upon receipt of said access command, enqueuing a predetermined access task in response to said access command,

(d) an access task enqueued being resultant in, in response to an access command, using said hashed query fragment to access data according to a hash table and returning a plurality of object identifiers corresponding to said accessed data.

(e) Said computer database system, upon receipt of said plurality of object identifiers, transmitting an object command for each said object identifier.

(f) Said computer database system, upon receipt of said object command, enqueuing a predetermined object task in response to said object command,

(g) an object task enqueued being resultant in, in response to an object command, using said object identifier to access data according to an object table and returning the location of the object, the plurality of object features of said said object and auxiliary information associated with the said object, corresponding to said accessed data.

44. The method of claim **43** 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.

45. The method of claim **44** wherein there are two query levels.

46. The method of claim **45** wherein said query node returns a plurality of locations of objects and auxiliary data in response to a predetermined query level.

47. [Version A] 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, fragmenting each said object feature of said plurality of object features into a plurality of object fragments, hashing each said object fragment of said plurality of object fragments into a hashed object fragment having a first portion and a second portion, transmitting an insert object message containing the location of the said object, the said plurality of 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 fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed object fragment,

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

(h) said query node, upon receipt of said insert message, using said second portion of said hashed object fragment to store data according to a local hash table located on said query node.

48. [Version B] A computer database system for storage and retrieval of information.

(a) Said computer database system, upon receiving a command from a user, enqueuing a predetermined task in response to said command,

(b) an insert task enqueued, in response to an insert command from said user, selecting a unique object identifier for the object contained in said insert command, extracting a plurality of features from an object contained in said insert command, fragmenting each said object feature of said plurality of object features into a plurality of object fragments, hashing each said object fragment of said plurality of object fragments, transmitting an insert object command containing the location of the said object, the said plurality of object features and auxiliary information about the said object, and transmitting an insert command for each hashed object fragment.

(c) Said computer database system, upon receipt of said insert object command, enqueuing a predetermined insert object task in response to said insert object command,

(d) an insert object task enqueued being resultant in, in response to an insert object command, using said object identifier to store the location of the said object, the said plurality of object features and auxiliary information about the said object in an object table.

(e) Said computer database system, upon receipt of said insert command, enqueuing a predetermined insert task in response to said insert command,

(f) an insert task enqueued being resultant in, in response to an insert command, using said hashed object fragment to store data according to a hash table.

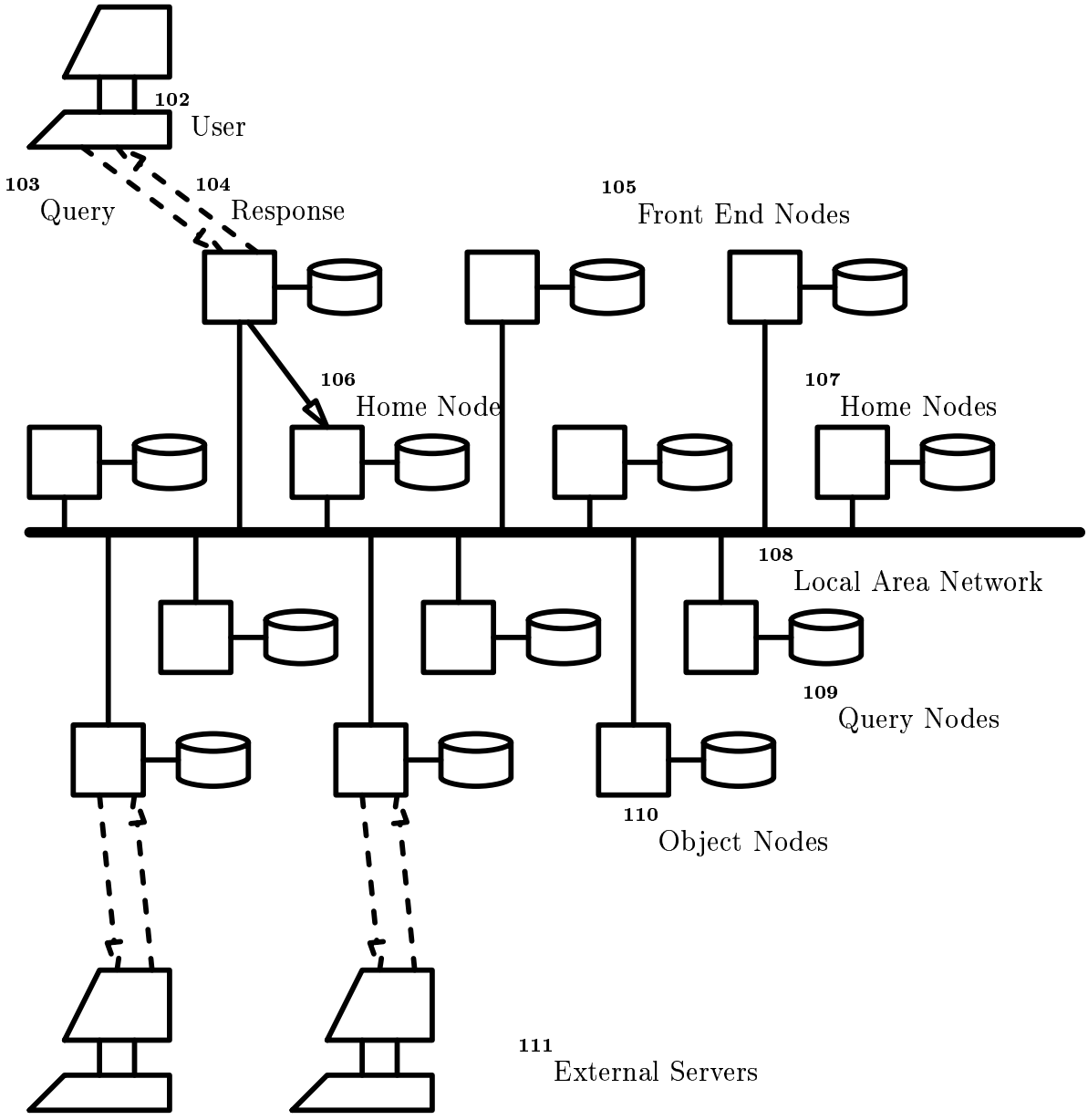
102
User
103
Query
104
Response
105
Front End Nodes
106
Home Node
107
Home Nodes
108
Local Area Network
109
Query Nodes
110
Object Nodes
111
External Servers

FIG. **1**

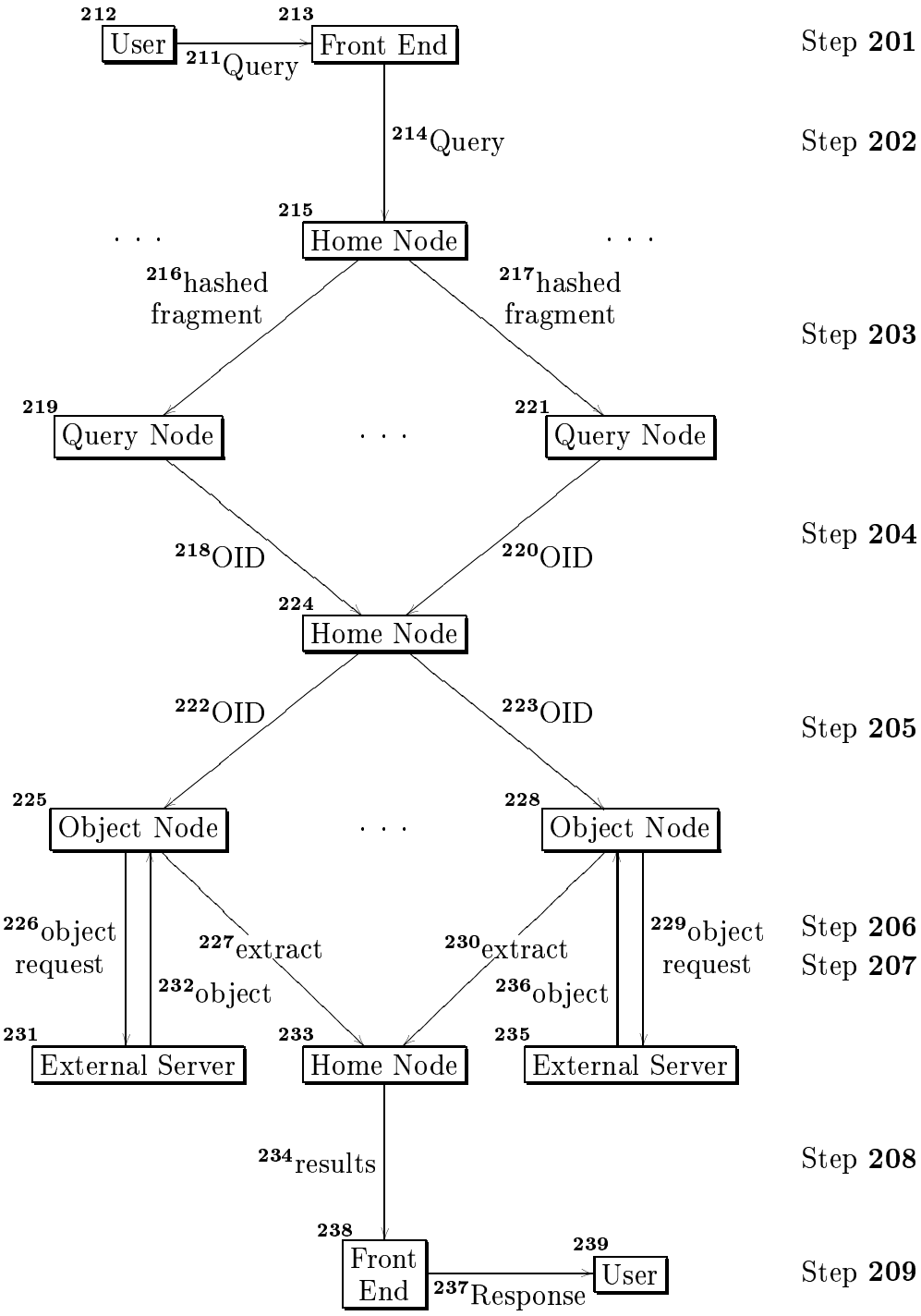
212
User
213
Front End
211Query
Step 201
214Query
Step 202
215
Home Node
216hashed fragment
217hashed fragment
Step 203
219
Query Node
221
Query Node
218OID
220OID
Step 204
224
Home Node
222OID
223OID
Step 205
225
Object Node
228
Object Node
226object request
227extract
230extract
229object request
Step 206
Step 207
232object
236object
231
External Server
233
Home Node
235
External Server
234results
Step 208
238
Front End
239
User
237Response
Step 209

FIG. **2**

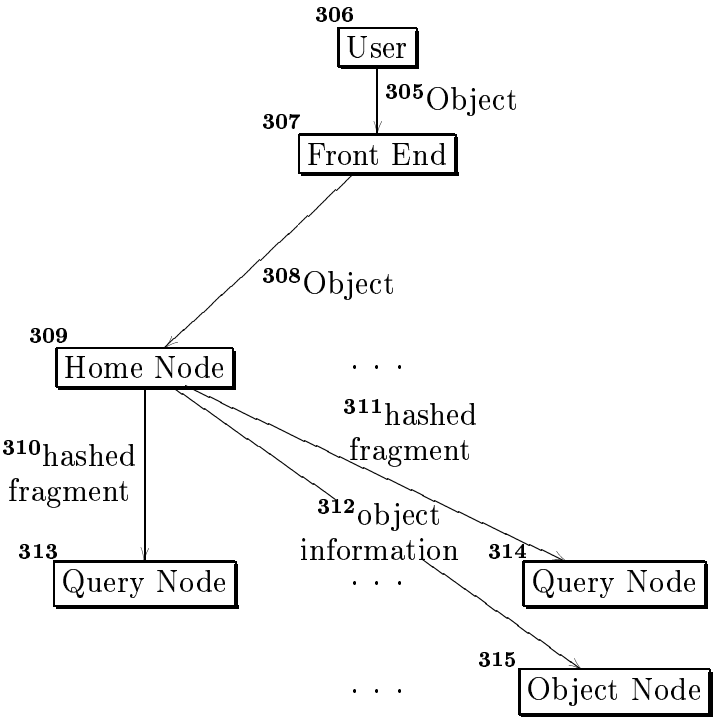

FIG. **3**

Query Message

402 Header	403 QID	404 HQF	405 Value

FIG. **4a**

Query Response Message

406 Header	407 QID	408 OID	409 Weight

FIG. **4b**

Object Message

410 Header	411 QID	412 OID	413 TS

FIG. **4c**

Object Response Message

414 Header	415 QID	416 OID	417 Location
418 Features...			
419 ...			

FIG. **4d**

Insert Message

420 Header	421 OID	422 HQF	423 Value

FIG. **4e**

Insert Object Message

424 Header	425 OID	426 TS	427 Location
428 Features...			
429 ...			

FIG. **4f**