Knowledge Extraction 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 data warehouse and data mining engine which indexes objects including images, sound and video streams, as well as plain and structured text.

An object from an external database is downloaded by a node, termed the warehousing node. The warehousing node extracts some features from the object 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 object to perform a search on its respective partition of the database. The results of the searches of the local databases are gathered by the warehousing node. The warehousing node uses these results to determine whether the object has already been indexed in the data warehouse. The warehousing node then extracts all the features from the object 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 object to store the feature in its respective partition of the database.

A pattern query is a search for a pattern in the data. A pattern query from a user is transmitted to one of the front end computers which forwards the pattern query to one of the computer nodes, termed the home node, of the data mining engine. The home node decomposes the pattern query into query steps, each step consisting of an object feature and a computation. The computation may involve additional query steps. The home node hashes the features. Each query step is transmitted to one node on the network, according to the hashed feature. Each node on the network which receives a query step uses the hashed feature of the query step to perform a search on its respective partition of the database, and the accessed data is used by the computation of the query step. If the computation of a query step contains additional query steps, then the additional query steps are evaluated recursively, and the data obtained by the recursive evaluation is used by the computation of the query step. The results of the searches of the local databases and the results of any recursive evaluations are gathered by the home node. The results of the pattern query are determined by the home node and returned 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] 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] M. Hurwicz. Take your data to the cleaners. *Byte Magazine*, January 1997.

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

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

[8] S. Weiss and N. Indurkhya. *Predictive Data Mining: A Practical Guide.* Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1998.

[9] J.-L. Weldon and A. Joch. Data warehouse building blocks. *Byte Magazine*, January 1997.

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

Organizations routinely collect large amounts of data on their customers, products, operations and business activities. Insights buried in this data can contribute to marketing, reducing operating costs and strategic decision-making. For example, if there is a strong correlation between the customers who buy one product with those

who buy another product, then those customers who have bought just one of them might be good prospects for buying the other product.

Analytical processing of data is primarily done using statistical methods to extract correlations and other patterns in the data. This kind of processing has been variously called *data mining*, *knowledge discovery* and *knowledge extraction*. A search for a specific pattern or kind of pattern in a large collection of data will be called a *pattern query*.

Large enterprises maintain many databases, many of which are transactional databases. The requirements of these databases are often in conflict with the requirements of data mining. Transactional databases are updated using small transactions operating in real time. Data mining, on the other hand, uses large pattern queries that do not have to take place in real time. To resolve this conflict, it is now common for data from a variety of sources to be downloaded to a centralized resource called a *data warehouse*.

The downloading and centralizing of data from diverse sources requires a number of tasks. The data must be extracted from the sources, transformed to a common, integrated data model, cleansed to eliminate or correct erroneous or inaccurate data and integrated into the central warehouse. In addition, one must ensure that every instance of every business entity, such as a customer, product or employee, has been correctly identified. This is known as the problem of *referential integrity*. All of these are difficult tasks, especially ensuring referential integrity when the data is being downloaded from databases that identify the business entities in slightly different ways. Current technology downloads data to the data warehouse as an independent activity from data mining. In contrast with data mining, for which there is a large research literature and many commercial products, data warehousing does not have a strong theoretical basis and few good commercial products.

Because data warehouses integrate many diverse data sources, it is necessary to specify an integrated data model for the data warehouse as well as a data mapping that extracts, transforms and cleanses data from each data source. Experience has shown that richer data models, such as object-oriented data models, are better suited for defining such an integrated data model and for defining the data mappings, than more limited data models, such as the relational model. Yet most data warehouses still use a flat record structure such as the relational model. Relational databases have a very limited data structure, so that synthesizing more complex data structures is awkward and error-prone. Some of the kinds of data that are poorly suited to storage in a relational database include: textual data in general, hypertext documents in particular, images, sound, multimedia objects and multi-valued attributes. Relational databases are also poorly suited for representing records that have a very large number of potential attributes, only a few of which are used by any given record.

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

information object is identified uniquely by an object identifier (OID). Each information object can have features, and some features can have an associated values. Information objects can also contain or refer to other information objects.

To assist in finding information in a 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, so the ones that are supported must be chosen carefully. For transactional databases, the workload is usually well understood, so it is possible to choose the indexes so as to optimize the performance of the database. For a data warehouse, there is usually no well defined workload, so it is much more difficult to choose which attributes to index.

3 Summary of the Invention

The present invention combines the two activities of data warehousing and data mining, thereby improving the basis and support for data warehousing. The term *knowledge extraction* will be used for the integration of the data warehousing and data mining activities. The present invention supports the indexing of sparse records which have large numbers of potential attributes, only a few of which are used in a particular record. The present invention supports the indexing of very large numbers of attributes in a uniform data structure, making it much easier to determine the workload characteristics necessary for achieving high performance.

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 knowledge extraction engine, simultaneously supporting both the data warehousing activity and the data mining activity.

First consider the data warehousing activity. The downloading of objects from another database to the warehouse is performed by a warehousing node. For an object downloaded from another database, the warehousing node must first determine whether the object might already be represented in the data warehouse due to a download from another database. If this might be the case, the warehousing node extracts some of the features of the object and then hashes these features. A portion of each hashed feature is used by the warehousing node as an addressing index by

which the warehousing node transmits the hashed object feature to a node on the network.

Each node on the network which receives a hashed object feature uses the hashed object feature to perform a search on its respective database. Nodes finding data corresponding to the hashed object feature return the OIDs of the warehoused objects possessing this feature. Such OIDs are then gathered by the warehousing node and a similarity function is computed. The similarity function is used to determine whether the object is already stored in the data warehouse. If the object is determined to be represented in the data warehouse, then the OID of the warehoused object is used for the downloaded object. If it is not already represented, then a unique OID is chosen for the object.

The warehousing node then extracts all of the features of the object and then hashes these features. A portion of each hashed feature is used by the warehousing node as an addressing index by which the warehousing node transmits the hashed object feature to a node on the network where the feature is stored in the data warehouse.

Next consider the data mining activity. A user wishing to search for a pattern in the data transmits a pattern query to one of the front end computers which in turn forwards the pattern query to one of the computer nodes of the network. The node receiving the pattern query, termed the home node of the data warehouse, decomposes the pattern query into query steps. A query step consists of a feature and a computation which may include additional query steps. The home node then hashes the features of the query steps. A portion of each hashed feature is used by the home node as an addressing index by which the home node transmits the query step to a node on the network.

Each node on the network which receives a query step uses the hashed query step feature to perform a search on its respective database. Nodes finding data corresponding to the hashed query step feature, perform the computation specified in the query step. If the computation does not contain any additional query steps, then the results of the computation are returned to the home node. If the computation does contain additional query steps, then the node takes the role of the home node with respect to the query steps contained in the computation. In particular, the node hashes the features of the contained query steps and transmits the query steps to other nodes. This process continues recursively until the computation is complete and the final results are returned to the original home node.

Upon receiving the results of the computation, the home node performs any remaining data aggregation specified by the original pattern query and transmits the information to the front end node. The front end node formats the response to the user, and 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 download information from another source to the data warehouse;

FIG. 3 is an overview of the steps used by the embodiment of the distributed computer database system to respond to a pattern query.

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 data warehouse and data mining 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 data warehouse may be of several types, including index nodes and warehousing nodes. The nodes of the data warehouse need not represent distinct computers. In one embodiment, the data warehouse consists of a single computer which takes on the roles of all index nodes and warehousing nodes. In another embodiment, the data warehouse consists of separate computers for each index node and warehousing 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 downloading of an object first, and referring also to FIG. **2**, in one embodiment objects are downloaded (Step **201**) by one or more warehousing nodes.

If an object may already be represented in the data warehouse due to a download from another database, the warehousing node extracts some of the features of the object which have been determined to be useful for this purpose, according to the integrated data model of the data warehouse.

A variety of feature extraction techniques can be used. For traditional field values, the possible values are partitioned into a collection of contiguous, non-overlapping ranges. Partitioning field values in this way is called *discretization*. Note that membership in one of these ranges is indexed; the field value itself is not indexed. If a different discretization must be used, then the field values must be indexed again using the discretization. Both discretizations may be used at the same time, or the new discretization can replace the other.

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

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

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

The warehousing node encodes each feature of the object 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. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for an object assures that data is distributed uniformly over the index nodes of the data warehouse during the storing of data.

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 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, the hashed object features are distributed (Step **202**) as probes to certain index nodes of the data warehouse, as determined by the first portion of the hash value.

The index nodes whose probes match the hashed features by which the data was initially stored on that index node respond to the query step by transmitting (Step **203**) the OIDs matching the hashed features of the requested information to the warehousing node. Thus all matches between the hashed probes and the local hash table of hashed features are returned or gathered to the warehousing node which initially hashed the object features.

The warehousing node then determines whether one of the OIDs represents the same object as the object to be warehoused. This determination is made by the warehousing node by comparing the degree of similarity between the object to be warehoused and the objects whose OIDs were returned. In one embodiment the measure of similarity is determined by

1. the features that are common to the objects, and

2. the features of the object to be warehoused that are not features of the object whose OID was returned.

This measure of similarity is based on the Feature Contrast Model of Tversky. The first term contributes a positive number to the similarity value, while the second term has a negative contribution. In addition the second term is multiplied by a predefined constant such that a feature in the second set has less effect on the similarity than one in the first set.

If the object is determined to be represented in the data warehouse, then an OID is already available for the object. If it is not already represented, then a unique OID is chosen for the object.

The warehousing node then extracts all of the features of the object according to the integrated data model of the data warehouse. The feature extraction techniques were discussed above. The warehousing node encodes each feature of the object by using a predefined hashing function as discussed above. 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 (Step **204**), and a second portion which is the local index value which is used to determine where data is to be stored at the index node (Step **205**).

Considering next the processing of a pattern query, and referring also to FIG. **3**, in one embodiment when a user transmits (Step **301**) a pattern query from the user

computer, the front end computer receives the pattern query. The front end computer is responsible for establishing the connection with the user computer to enable the user to transmit a pattern 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 pattern query is acceptable, the front end computer performs any reformatting necessary to make the pattern query compatible with the requirements of the data warehouse. The front end computer then transmits the pattern query to one of the index nodes of the data warehouse (Step **302**), which is then defined as the home node of the data warehouse for that pattern query.

The home node decomposes the pattern query into a series of query steps. Each query step consists of a feature and a computation. The computation determines what action the query step is to perform. The most common computations are statistical functions that aggregate information stored in the data warehouse. The computation may contain additional query steps.

For each query step, the home node encodes the feature by using a predefined hashing function as described above. The hashed feature and query step are transmitted by the home node to an index node (Step **303**) using the hashed feature as described above.

Index nodes whose hashed features match the index features by which the data was initially stored on that index node respond to the query step by retrieving data in the local hash table of index terms that match the hashed feature and by performing the computation specified in the query step. If the computation contains any additional query steps, then the index node acts as the home node for a new pattern query, called a *subquery*, which is processed as described above (Step **304**). Whether the computation contains additional query steps or not, the index node returns the results of its computation to the home node of the query step that it received (Step **305**).

When the results of all the query steps of the original pattern query have been received, the home node performs any data aggregation specified by the original pattern query and returns the pattern information to the user. In one embodiment the returned pattern information is transmitted to the front end (Step **306**) computer which formats the response appropriately and transmits the response to the user (Step **307**). 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 message formats used in the preferred embodiment, refer to FIG. **4**. The Warehouse Message four fields: Header, Object Identifier (QID), Hashed Object Fragment (HOF) and Value. The Header field specifies that this message is a Warehouse Message and also specifies the destination index node. The destination

index node is determined by the first portion of the hashed object fragment. The OID field contains an object type specifier and an object identifier. The HOF field contains a fragment type specifier and the second portion of the hashed object fragment produced by the Hashing Module. The Value field contains an optional value associated with the fragment. The fragment type specifier determines whether the Warehouse Message contains a Value field, and if the Warehouse Message does contain a Value field then the fragment type specifier determines the size of the Value field.

The Warehouse Response Message has two parts: Identifier and Values. The Identifier part has four fields: Header, OID1, OID2 and Weight. The Header field specifies that this message is a Warehouse Response Message and also specifies the destination warehouse node. The destination warehouse node is the warehouse node from which the corresponding Warehouse Message was received. The two OID fields contain an object type specifier and an object identifier. The first OID field is the same as the OID field of the corresponding Warehouse Message. The second OID field identifies an object that has been previously indexed. The Weight field contains an optional weight associated with the object identified by OID1. The object type specifier of OID1 determines whether the Warehouse Response Message contains a Weight field, and if the Warehouse Response Message does contain a Weight field then the object type specifier of OID1 determines the size of the field. The Values part of the Warehouse Response Message contains data associated with the object identified by OID2. The structure and size of the Values part is determined by the object type specifier of OID2.

The Insert Message has four fields: Header, OID, HOF and Value. The Header field specifies that this message is an Insert Message and also specifies the destination index node. The destination index node is determined by the first portion of the hashed object fragment. The OID field contains an object type specifier and the object identifier. The HOF field contains a fragment type specifier and the second portion of the hashed object fragment produced by the Hashing Module. The Value field contains an optional value associated with the fragment. The fragment type specifier determines whether the Insert Message contains a Value field, and if the Insert Message does contain a Value field then the fragment type specifier determines the size of the Value field.

The Query Step Message has two parts: Identifier and Subqueries. The Identifier part has four fields: Header, Query Step Identifier (QSID), Hashed Query Fragment (HQF) and Value. The Header field specifies that this message is a Query Step Message and also specifies the destination index node. The destination index node is determined by the first portion of the hashed query fragment. The QSID field contains a query type specifier and a query step identifier. The HQF field contains a fragment type specifier and the second portion of the hashed query step 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 Step Message contains a Value field, and if the Query Step Message does contain a Value field then the fragment type specifier determines the size of the Value field. The Subqueries part of the Query Step Message contains a number of subqueries. A Query Step Message having no subqueries is called a Simple Query Step Message.

The Query Step Response Message has two parts: Identifier and Values. The Identifier part has two fields: Header and QSID. The Header field specifies that this message is a Query Step Response Message and also specifies the destination index node. The destination index node is the same as the the index node from which the corresponding Query Step Message was received. The QSID field contains a query type specifier and a query step identifier. The Values part of the Query Step Response Message contains the result data of the query step. The structure of the Values part is determined by the query type identifier.

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

Considering next the modules contained in the warehousing nodes used in the preferred embodiment, refer to Fig. 5. The Downloader scans external databases to download objects for warehousing and indexing by the knowledge extraction engine. Each warehousing node may have a different Downloader module. For example, the Downloader can download data from relational databases using a standard SQL protocol such as ODBC or a proprietary protocol defined by one of the relational database vendors. Downloading in this case is performed using one or more SQL queries. For another example, the Downloader can be an Information and Content Exchange (ICE) subscriber that negotiates to obtain content from syndicators over the Internet. This is the preferred mechanism for obtaining time-sensitive content such as news feeds. The Downloader transfers objects to the Feature Extractor.

The Feature Extractor extracts features from an object. If the object is a relational database record, then feature extraction includes steps such as selecting the fields that will be indexed, reformatting fields and eliminating or correcting data that is determined to be erroneous. 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. Features are indexed by using a number of Insert Messages.

The Feature Extractor must also map each object identifier in an external database to an object identifier of the knowledge extraction engine. Each external databases can have its own mechanism for assigning object identifiers, and features of the same object may be stored in each external database with a different object identifier. For example, one external database might use a social security number. Another external database might use an employee identifier. The mapping from external object identifier to is achieved by using a number of Warehouse Messages.

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. For an object consisting of a relational database record, the features are the attributes that were selected, reformatted and corrected by the Feature Extractor. 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 Warehouse Message or an Insert Message to the Communication Module, depending on whether the purpose of the fragment is to achieve an object identifier mapping or to index an object feature, respectively.

The Similarity Comparator receives Warehouse Response Messages and produces Insert Messages which are transferred to the Communication Module. The Similarity Comparator gathers all the warehouse responses for an object whose identifier is being mapped. For each object in the responses, the Similarity Comparator determines the relevance of each object identifier returned in the search. This determination of relevance is made by the warehousing node by comparing the degree of similarity between the object whose identifier is being mapped 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. If a compatible OID is found, then the OID will be used as the mapped object identifier, and the OID is transferred to the Feature Extractor. If no compatible OID is found, then a new object identifier is chosen and transferred to the Feature Extractor.

Considering next the modules contained in the index nodes used in the preferred embodiment, refer to Fig. 6. The Fragment Table receives Warehouse Messages, Insert Messages and Simple Query Step Messages. In the case of a Warehouse Message the Fragment Table retrieves an entry in the local hash table using the hash value in the

HOF field. The type specifier in the HOF field and the entry in the local hash table are transferred to the Fragment Comparator. In the case of a Simple Query Step Message the Fragment Table retrieves an entry in the local hash table using the hash value in the HQF field. The entry in the local hash table are returned to the Query Processor using a Query Step Response Message. 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 HOF 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 Warehouse Response Message.

The Query Parser parses a pattern 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 a query step. A query step can have a number of subqueries. Each subquery is specified using a query computation tree. The query computation tree is transferred to the Query Processor.

The Query Processor is responsible for administering the processing of pattern queries. Upon receiving a query computation tree from the Query Parser, it assigns a query identifier (QID) to the query, and it assigns a query step identifier (QSID) to each leaf node that specifies a query step. A query step that has no subqueries is called a simple query step. A query step is processed by transmitting a Query Step Message to the specified index node by means of the Communication Module. The Query Processor at the specified destination index node processes the Query Step Message by transferring a Simple Query Step Message to the Fragment Table which responds with a Query Step Response Message. The Query Processor then sends the Query Step Response Message to the index node that originally sent the Query Step Message. As a result, Query Processor modules both send and receive Query Step Messages and Query Step Response Messages. As Query Step Response Messages are received, the processing specified in the query computation tree is performed. When a query step has a subquery, the subquery requires the processing of additional query steps. When the entire pattern query has been computed, the results are formatted and transmitted to the front end from which the pattern query was received. For example, the results may be given as a graph or table.

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 of warehousing objects or locations of objects in a manner which is conducive to knowledge extraction using pattern queries in a distributed computer database system having a plurality of index nodes and a plurality of warehousing nodes connected by a network, said method comprising the steps of:

   (a) extracting, by a warehousing node, a first plurality of features from an object downloaded from another database;

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

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

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

   (e) returning, by each said index node accessing data according to said respective hashed object feature, a plurality of object identifiers corresponding to said accessed data to said warehousing node;

   (f) determining, by said warehousing node, whether the said object is to be assigned an object identifier from the said plurality of object identifiers, or the said object is to be assigned an object identifier that is not yet in use;

   (g) assigning, by said warehousing node, an object identifier to the said object according to the said determination;

   (h) extracting, by said warehousing node, a second plurality of features from said object;

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

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

(k) 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.

2. The method of claim **1** further comprising the step of determining, by said warehousing node, a measure of similarity between said accessed data and said object; subsequent to the step of returning said first plurality of object identifiers.

3. The method of claim **2** 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 object; and

(b) features possessed only by the said object.

4. A method for data mining using pattern queries in a distributed computer database system having a plurality of index nodes connected by a network, said method comprising the steps of:

(a) selecting a first one of said plurality of index nodes, herein termed the home node of the pattern query;

(b) extracting, by said home node, a plurality of query steps from a pattern query by a user, each said query step consisting of a feature, a plurality of query steps and a computation specification;

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

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

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

(f) recursively evaluating, by said index node, each query step of said plurality of query steps contained in said respective query step transmitted by said home node, said index node acting as the home node of said query step of said plurality of query steps;

(g) computing, by said index node, pattern information according to said computation specification of said respective query step transmitted by said home node, according to said accessed data and pattern information determined by said recursive evaluation of each said query step of said plurality of query steps contained in said respective query step transmitted by said home node;

(h) returning, by each said index node, said pattern information to said home node.

5. The method of claim **4** further comprising the step of receiving, at said home node, said pattern query from said user, prior to the step of extracting query steps from said pattern query.

6. A distributed computer database system for warehousing of information objects or locations of information objects, comprising

(a) a plurality of warehousing nodes; and

(b) a plurality of index nodes;

(c) said plurality of warehousing nodes and said plurality of index nodes connected by a network,

(d) wherein each said warehousing node, upon downloading an object, extracts a first plurality of features from said object, hashes each said object feature of said first 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) wherein each said index node uses said second portion of said hashed object feature to access data according to a local hash table located on said index node, returning a plurality of object identifiers corresponding to said accessed data to said warehousing node,

(f) wherein said warehousing assigns to said object either one of said object identifiers of said plurality of object identifiers or an object identifier that is not yet in use, extracts a second plurality of features from said object, hashes each said object feature of said second 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,

(g) 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.

7. The distributed computer database system of claim **6** wherein said warehousing node determines a measure of similarity between said accessed data and said object for use in assigning an object identifier to said object.

8. The method of claim **7** wherein said warehousing node measures similarity using a similarity function determined by:

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

(b) features possessed only by the said object.

9. A distributed computer database system having a data mining tool for handling pattern queries from a user comprising:

(a) a plurality of index nodes;

(b) said plurality of index nodes connected by a network.

(c) wherein each said index node, upon receiving a pattern query from a user, and termed the home node of said pattern query, extracts a plurality of query steps from said pattern query, hashes the feature contained in each said query step of said plurality of query steps into a hashed query step feature having a first portion and a second portion, and transmits each said hashed query step feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed query step feature,

(d) further wherein each said index node uses said second portion of said hashed query step feature to access data according to a local hash table located on said index node, recursively evaluates each query step contained in said respective query step, computes pattern information according to said accessed data and pattern information determined by said recursive evaluation, and returns said pattern information to said home node.

10. A distributed computer database system for warehousing and data mining, comprising:

(a) a plurality of warehousing nodes; and

(b) a plurality of index nodes;

(c) said plurality of warehousing nodes and said plurality of index nodes connected by a network,

(d) each said warehousing node, upon receiving a download command, enqueuing a predetermined task in response to said download command,

(e) a download task enqueued, in response to a download command, extracting a first plurality of features from an object contained in said download command, hashing each said object feature of said first plurality of object features into a hashed object feature having a first portion and a second portion, and transmitting a retrieve 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 retrieve message, using said second portion of said hashed object 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 warehousing node,

(g) said warehousing node, upon receipt of said plurality of object identifiers from said plurality of index nodes, assigning to said object either one of said object identifiers of said plurality of object identifiers or an object identifier that is not yet in use, extracting a second plurality of features from said object, hashing each said object feature of said second 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,

(h) 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.

11. The distributed computer database system of claim **10** wherein said warehousing node determines a measure of similarity between said accessed data and said object for use in assigning an object identifier to said object.

12. The method of claim **11** wherein said warehousing node measures similarity using a similarity function determined by:

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

(b) features possessed only by the said object.

13. A distributed computer database system having a data mining tool for handling pattern queries from a user, comprising:

    (a) a plurality of index nodes;

    (b) said plurality of index nodes connected by a network.

    (c) each said index node, upon receiving a command from a user, said index node termed the home node of the command, enqueuing a predetermined task in response to said command,

    (d) a pattern query task enqueued being resultant in, in response to a pattern query command from said user, extracting a plurality of query steps from a pattern query contained in said pattern query command, hashing the feature contained in each said query step of said plurality of query steps into a hashed query step feature having a first portion and a second portion, and transmitting a query step message containing each said hashed query step feature to a respective one of said plurality of index nodes indicated by said first portion of said hashed query step feature,

    (e) said index node, upon receipt of said query step message, using said second portion of said hashed query step feature to access data according to a local hash table located on said index node recursively evaluating each query step contained in said respective query step, computing pattern information according to said accessed data and pattern information determined by said recursive evaluation, and transmitting a message returning said pattern information to said home node.

14. The method of claim **13** wherein said pattern query message requests predetermined data from said index node in response to a pattern query contained in said pattern query command from said user.

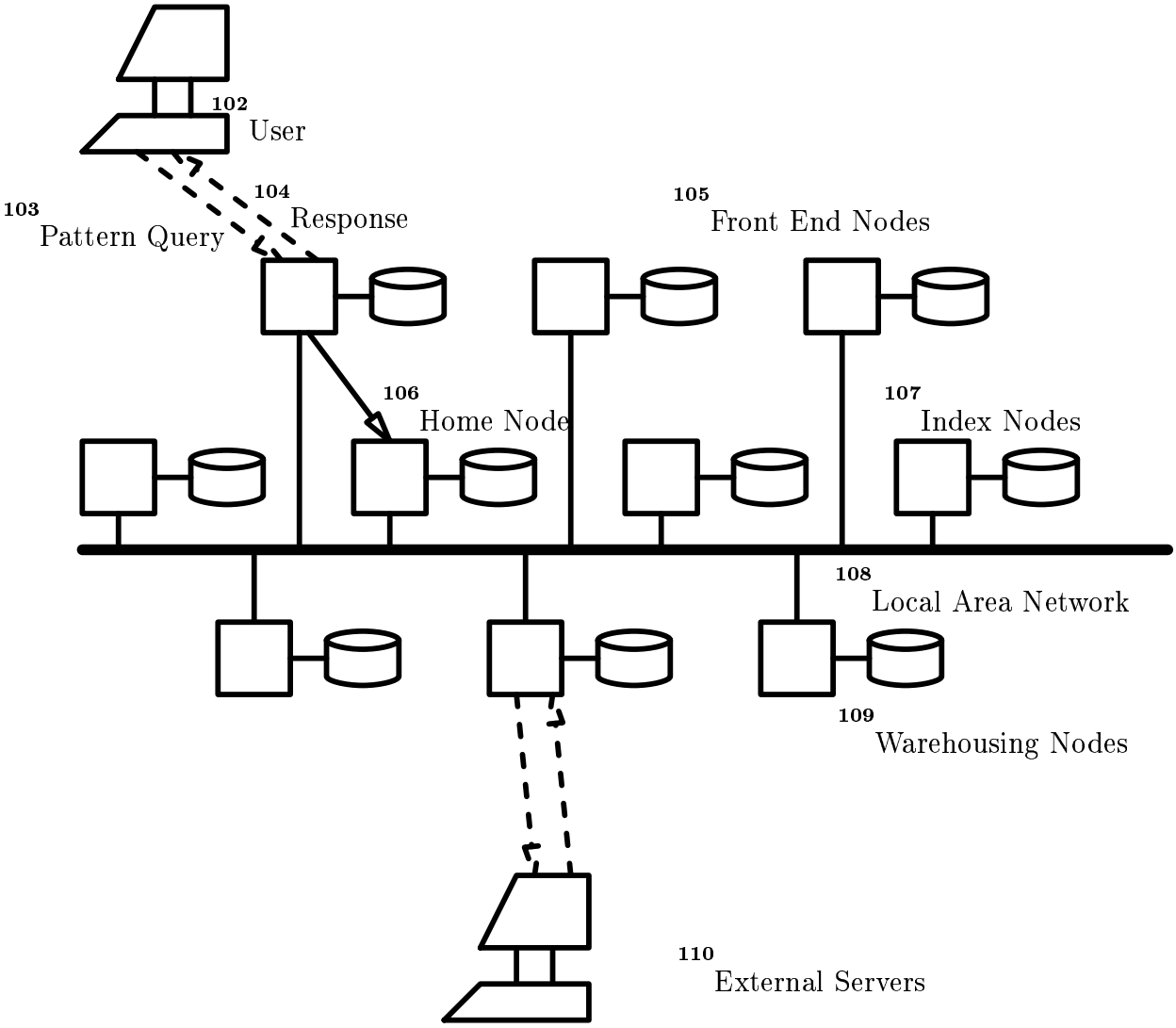
102
User
103
Pattern Query
104
Response
105
Front End Nodes
106
Home Node
107
Index Nodes
108
Local Area Network
109
Warehousing Nodes
110
External Servers

FIG. **1**

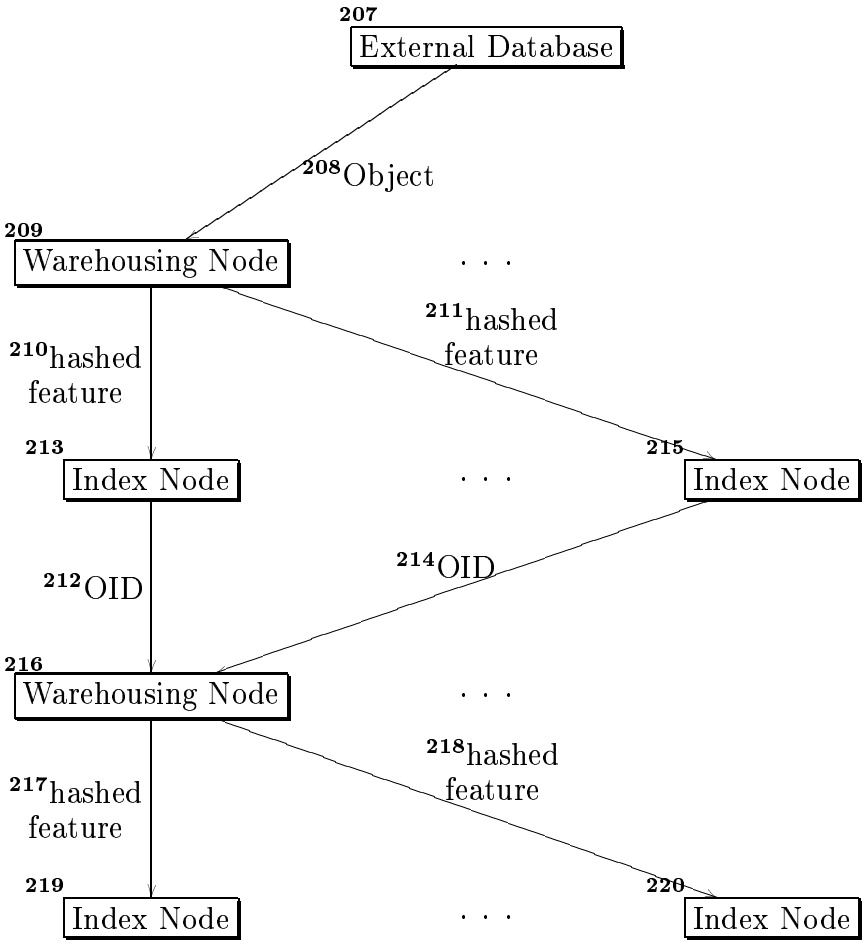
207
External Database
208 Object
(Step 201)
209
Warehousing Node
. . .
211 hashed feature
210 hashed feature
(Step 202)
213
Index Node
. . .
215
Index Node
212 OID
214 OID
(Step 203)
216
Warehousing Node
. . .
218 hashed feature
217 hashed feature
(Step 204)
219
Index Node
. . .
220
Index Node
(Step 205)

FIG. **2**

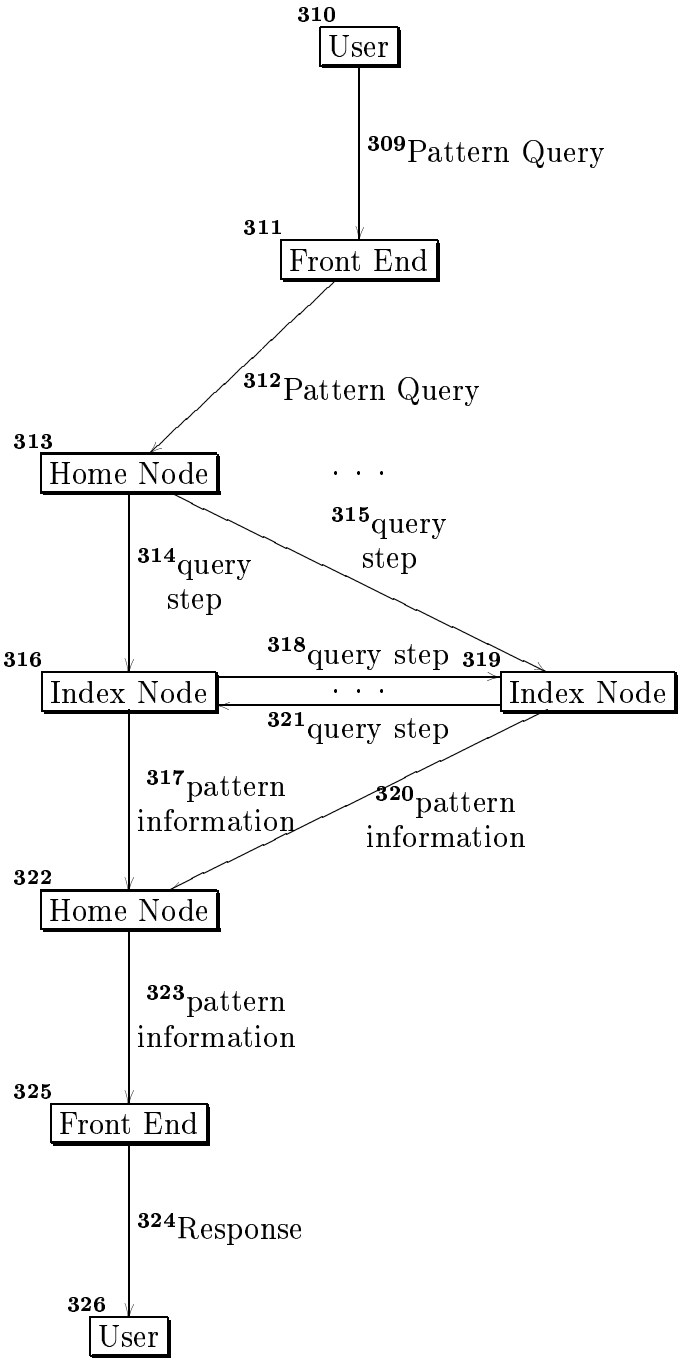
310
User
309 Pattern Query
(Step 301)
311
Front End
312 Pattern Query
(Step 302)
313
Home Node
. . .
315 query step
314 query step
(Step 303)
316
318 query step
319
Index Node
. . .
Index Node
(Step 304)
321 query step
317 pattern information
320 pattern information
(Step 305)
322
Home Node
323 pattern information
(Step 306)
325
Front End
324 Response
(Step 307)
326
User

FIG. **3**

Warehouse Message

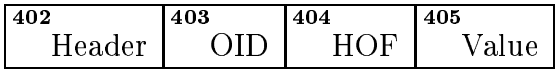

FIG. **4a**

Warehouse Response Message

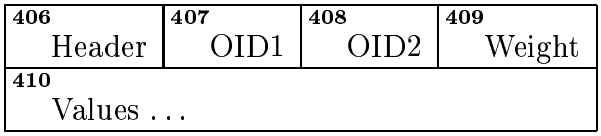

FIG. **4b**

Insert Message

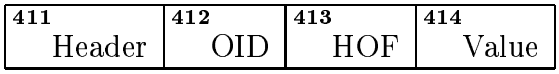

FIG. **4c**

Query Step Message

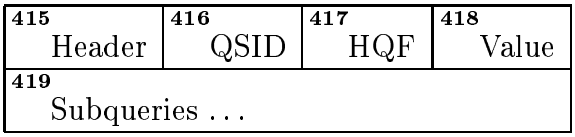

FIG. **4d**

Query Step Response Message

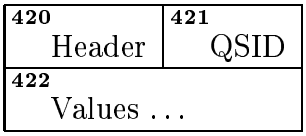

FIG. **4e**