A New Feasible Approach to Natural Language Database Query

by

Veera Boonjing and Cheng Hsu***

February, 2004

To appear in
International Journal on Artificial Intelligence Tools, 2005

*Kbveera@kmitl.ac.th, (662) 3269982, Mathematics and Computer Science
King Mongkut's Institute of Technology Ladkrabang, Ladkrabang, Bangkok
10520 Thailand

**Hsuc@rpi.edu, (518) 276-6847, Decision Sciences and Engineering
Systems, Rensselaer Polytechnic Institute, Troy, NY 12180-3590
(Please send correspondences to Cheng Hsu.)

Abstract

A truly natural language interface to databases also needs to be practical for actual implementation. We developed a new feasible approach to solve the problem and tested it successfully in a laboratory environment. The new result is based on metadata search, where the metadata grow in largely linear manner and the search is linguistics-free (allowing for grammatically incorrect and incomplete input). A new class of reference dictionary integrates four types of enterprise metadata: enterprise information models, database values, user-words, and query cases. The layered and scalable information models allow user-words to stay in original forms as users articulated them, as opposed to relying on permutations of individual words contained in the original query. A graphical representation method turns the dictionary into searchable graphs representing all possible interpretations of the input. A branch-and-bound algorithm then identifies optimal interpretations, which lead to SQL implementation of the original queries. Query cases enhance both the metadata and the search of metadata, as well as providing case-based reasoning to directly answer the queries. This design assures *feasible solutions* at the termination of the search, even when the search is incomplete (i.e., the results contain the correct answer to the original query). The necessary condition is that the text input contains at least one entry in the reference dictionary. The sufficient condition is that the text input contains a set of entries corresponding to a complete, correct single SQL query. Laboratory testing shows that the system obtained accurate results for most cases that satisfied only the necessary condition.

Categories and Subject Descriptors: H.2.3 [Database Management]: Languages – query languages; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – query formulation; H.5.2 [Information Interfaces and Presentation]: User Interfaces

General Terms: database query, natural language interface

Additional Keywords: metadata search, Metadatabase, reference dictionary

1. Asking the database in your own language

Talking to a computer in a natural language such as plain English is always a dream that drives the progress of human-computer interaction work. However, what is the value added for natural language ***database*** queries? Is it not true, for example, that ***direct*** database users are very content with using SQL, a menu-based interface, or the so-called 4th-generation languages to query their databases? We submit that a truly natural language interface for database query is always an important technology because of its promises. First, such a technology could ease the user's burden of obtaining a complete set of specific information required by traditional querying methods - e.g., database objects and values needed in the clauses of SQL statements. Imprecise query formulations almost always lead to no results or meaningless results. In contrast, we observed that "incomplete" (or, ambiguous) expressions of queries in English could in fact be sufficient for the proposed approach to produce correct answers. Second, natural language query would be necessary for indirect, end users to use databases directly without going through some intermediary professionals. That is, the technology could facilitate the (partial) automation of such systems as OnStar, a real time mobile information system to assist automobile drivers. This class of systems is gaining prominence as the Web Phone technology widens its application in the society. Finally, the results developed for database query might offer some fresh ideas to the general areas of natural language interface for computer. After all, many natural language applications are information retrieval in nature, regardless of their technical characteristics being traditional databases, documents, or knowledge bases.

The above promises predicate on having sufficient solutions in the field. However, available solutions tend to lack ***feasibility*** either in their naturalness or in scalability for actual applications. To illustrate natural language database query, we list below a number of actual queries from mocked end users against a laboratory Computer-Integrated Manufacturing (CIM) database that we tested in the research. Many of them were incorrect or incomplete for the English language, but could happen easily in actuality.

- Get billing address of John Smith.
- Get orders of John Smith
- Just give me everything we have about orders of John Smith and Jim Kowalski.
- I'd like to know customer names who order PZ1.
- PZ1 customers
- How about PZ1 orders?

- Look at John Smith’s orders, now give me order_id, part_id, work order quantity, and num_completed of his orders.
- Give all order_id and models that John Smith’s orders.
- Do we have records about John Smith, Jim Kowalski, and William Batt?; they are our clients.
- Not done orders
- Now I want to know cost of PZ1 and PZ10
- What models did John Smith order?
- Go to Tommy Lee's records of orders; list his order_id and order status.
- Make a list of orders that are not done yet. Just give me order no. and customer names, ok?
- Now give me the details of William Batt’s orders that are not done.
- Which parts of orders are milling and assembly.
- Show the details of order no. 00008 and which customers placed this order.
- Order 00010: what is its due date?
- Is Jim Kowalski’s billing address and shipping address the same?
- Find customer names whose orders are not done; please include each customer's billing address
- For all orders: what are the parts associated with order no. 00006?
- Part pz1, pz2, ba, and bb;
- William Batt's orders, please.
- Find all 'PZ3' order information made by Tommy Lee.
- I have asked you many times and I still have not heard a clear answer. Listen, I would like to know the status of my order. As I know, my order no is 00009. Please make sure to include part id and part status for each part in the order. Could you please help? Thanks.
- Step on John Smith’s orders. I want to know the exact status of all parts he ordered including their descriptions, costs, and quantities completed to-date. I understand that you're supposed to deliver it on 12/31/99. I know I might not have given you all the information you need, but please do your best, anyway. Okay? I'm waiting...
- How about part PZ1?
- All customers.

In these truly natural queries, not only multiple different interpretations exist for the same expressions, the interpretations might not even contain sufficient data to complete the query for processing on the database. Thus, the two basic problems facing natural language database query are ***ambiguities in intent and specification***, and both require the system to possess deep and broad metadata about the underlying database to resolve.

A significant portion of previous results in the field develops specific concept-form-syntax models either for the users to use explicitly (imposed templates) or for the applications to target implicitly (presumed patterns). These models provide precise interpretation and mapping to traditional database query languages when the user input matches exactly. Thus, they control the ambiguities at the expense of naturalness. We might classify these results into five categories according to the technical nature of their linguistic models. They include (1) the template-based approach [e.g., Weizenbaum, 1966; Wilks, 1978; Shankar and Yung, 2001], (2) the syntax-based approach [e.g., Wood, 1970, 1978; 1980; Adam and Gangopadhyay, 1997], (3) the semantics-grammar-based approach [e.g., Hendrix et. al., 1978; Waltz, 1978; Codd, 1978; Codd et. al., 1978; Templeton and Burger, 1986; Owei, 2000], (4) the intermediate-representation-language-based approach [e.g., Warren and Pereira, 1982; Brajnik, et.al., 1986; Bates et.al., 1986; Gross et.al., 1987; Wu and Dilts, 1992], and (5) the concept-based models such as MindNet [Richardson et. al., 1998], WordNet [Fellbaum, 1998], and conceptual dependency [Schank, 1973, 1975]. Some recent works combine these results [Metais 2002] and have also tried to offer more naturalness by narrowing the application domain and engaging more natural language forms and rules [e.g., Zue, 1999; Rosenfeld et.al., 2000; Shankar and Yung, 2001 – more in the References].

Another fundamental approach developed previously is to use a general linguistics-string model to allow for interpretation of free-text queries. A prevailing idea of such models is the n-gram-based dictionary, containing all possible uses (permutations) of words found in all past queries to interpret new queries. It then maps the interpretation to a semantic representation of the database structure and thereby determines the executable queries for the original natural language input. Examples include many approaches using relation, object, tree, space vector, probabilistic measures, semantic parsing, and other designs to relate concepts to strings [Maier, 1983; Maier and Ullman, 1983; Wald and Sorenson, 1984; Johnson, 1995; Meng and Chu, 1999; Zhang et. al., 1999; Kim 2000; Jarvelin et.al. 2001; and Mittendorfer and Winiwarter 2002]. Some other results limit the design to particular applications in order to control the size of the models [e.g., Janus 1986; Motro 1986, 1990; Shimazu et. al., 1992; Guida and Tasso 1982; Meng and Chu 1999; Zhang et. al. 1999; ask.com 2002 – more in the References]. A concern for these results is their scalability. When the linguistics used are simplistic, such as containing only synonyms to the database objects, the question of multiple interpretations - i.e., the ambiguity in intent - looms large. When, on the other hand, the models are theoretically comprehensive enough to deal with ambiguities, they may become too costly to implement for non-trivial applications. The size of an n-gram dictionary could increase exponentially as the number of queries and users increases.

In all these approaches, incomplete input could still cause ambiguities in specification and/or intent, if the systems do not possess sufficient metadata to "complete the picture" for the users. Furthermore, a truly natural language interface can never guarantee complete success for all uses at all times - even humans routinely misunderstand other humans. Thus, it has to be able to always yield a feasible solution that contains the correct answer plus some additional but relevant information, and to continuously improve its performance through experience: obtaining (online) users' responses and/or the cases on the users. It follows that interaction with users is always critical for the system to close the loop and assure its validity. However, to make the interaction meaningful, the system has to possess sufficient metadata in order to provide a useful reference scheme on which to base the interaction (e.g., dialogue). Previous results have shown the need for sufficient interaction capability to achieve the above goal [Mylopoulos et.al. 1976; Codd 1978; Codd et.al. 1978; Waltz 1978; Kaplan 1984; Carbonell 1983; Rissland 1984; Norcio and Stanley 1989; Miller 1975; Soderland 1997; Mooney 1997, 2001; and Ruber 2001].

Therefore, we submit that ***open***, ***deep and scalable metadata*** about both the structure and application of the databases is a key to solving the problem of natural language query [Boonjing and Hsu 2002, 2003]. With sufficient metadata, the system could resolve ambiguity in specification and support a reference dictionary capable of resolving ambiguity in intent that grows linearly. Moreover, with them, the system could develop an efficient interaction with the user to assure a feasible closure for query processing as well as to effect continuous improvement. We envision the following metadata: query cases as in case-based reasoning, an enterprise information model expandable to include any number of semantic layers, database objects and values, and open user-words recognized for the information model and database objects and values. We then develop a ***feasible region*** interpretation approach using metadata to achieve natural language database query.

We discuss the new approach in Section 2, its core method and algorithms in Section 3, and its possible implementation in Section 4. We then discuss its feasibility in Section 5 to substantiate the above claims. The discussion uses results from a laboratory testing of the approach to analyze the soundness and scalability of the basic logic of metadata search. Finally, we discuss the possible extension of the core method in the last section, Section 6.

2. Formulating the Metadata Search Approach

The four types of ***metadata***: ***query cases, enterprise information model, database objects and values, and user-words***, mutually support each other in an integrative manner. Database objects (structure, operators, etc.) and values are most straightforward, since they are what the database is and are necessary for specifying queries. Enterprise information models represent well-defined applications and the basic concepts of these applications for the underlying databases; thus, they represent database linguistics and support user linguistics. An information model could be flat, capturing only the direct database structure and hence becomes fundamentally not scalable - the case with most previous results. However, it could also include scalable layers of semantics on top of the database structure to enrich the linguistics. The more scalable and layered, the more completely this subset of metadata represents the concepts users refer to in their natural language queries. Users would use their own words to refer to these concepts and database objects and values when they articulate a request against the database. The individual phrases and words that users use originally (not permutations of the original words that the system derives and generates) are the user-words. Query cases are past examples of user queries and their correct results. They should include minimally the exceptions - i.e., the cases where the system would have to engage an interaction with the users to obtain the correct answers. Thus, a case could include user, application and other pertinent information to assist its future use in interpreting the queries from the same user, the same application and the like. In this sense, cases close the gap between the metadata collected and the ones required at any given point of time. The system could use cases to interpret natural queries directly, if necessary, in a case-based reasoning manner. Clearly, database values, user-words and cases would grow as the use of the database continues; but the growth is proportional linearly to the actual queries processed. Other types of metadata could grow, too, to a much less extent. The enterprise information model is at the core of the metadata. It allows the system to infer the missing information to complete the incomplete queries; and it minimizes the need for user-words. Together with cases, they make it possible for a sufficient reference dictionary to avoid using n-grams. All together, they promise a feasible solution to resolving the ambiguity in intent and specificity.

A natural query is reducible to the set of user-words, concepts, database objects and values it contains. Conversely, a particular combination of these keywords could represent a particular natural query. However, the set of keywords may not always correspond to a unique and complete database query that the system can process correctly. The reasons include possible multiple mappings of the keywords to database objects/values, and incomplete keywords that the systems can complete in multiple ways. This is where the ***search*** approach comes into play. We first store the metadata into a reference dictionary and manage it as a Metadatabase (see the next section), and then use networked graphs to represent their logical structure. An interpretation of a set of keywords (metadata contained in the reference dictionary that users use) is an image of the set on the logical structure. The system identifies implicitly all permissible interpretations (including the ones derived from the reference dictionary for incomplete input) for the set and evaluates their relative merits to optimize the interpretation for the query. We use the branch-and-bound method to conduct this search over the graphs to achieve implicit enumeration and evaluation. All interpretations identified and derived are complete in their information content for database query processing. Then, the optimal interpretation leads to a query using the underlying database query language (such as SQL). This approach is free of the need to understand linguistically the human meaning of the words and phrases used. The situation is fundamentally different from generic natural language processing where the possibility of permutation is virtually infinite. In this approach, the system only needs one keyword to start the search process, since it could infer possible (multiple) interpretations from the graphs of the logical structure that contain it. When, in the ideal case, the original user query contains a complete set of metadata free of ambiguity, such as the ones that satisfies an SQL programmer, no derivation is necessary and a solution is readily available. Among all possible interpretations for a set of keywords, the one that involves least addition of derived keywords (including zero) should be the one most accurate to the user's intent. The argument here is simple: users would not be deliberately wordy with respect to their own standards, nor misleading, when querying a database. Therefore, minimal sufficiency is a good search criterion while the logical structure is the constraint of the metadata search algorithms, in a constrained optimization paradigm.

In the final analysis, we integrate all four types of metadata through an extensible metadata representation method so that every piece of resources references to all other related resources for query interpretation. The core of the reference dictionary (enterprise information model and initial user-words) will be a design-time product; while query cases, additional user-words, changes to the information model, and database values will be added as the database system evolves. The reference dictionary-based logical structure holds four fundamental promises: it generates search-ready graphics; it supports case-based reasoning; it assures complete interpretations of natural queries; and it simplifies user-words. The last point is worth of further elaboration. Consider the theoretical complexity of processing only one natural query. Suppose the text consists of a stream of **n** words of which **m** are database objects or other recognized metadata. There would then be **n/m** words associated with each known term; these words become the candidate user-words for the term. The expected number of permutations of these user-words would be **m*(n/m)!** for the natural query. As **n** grows over time with new queries and applications, the number of user-words would grow in this rate - and this is the worst-case complexity of a linguistic dictionary. Obviously, the system wants to increase the number **m** (hits) since the bigger **m** is, the fewer (exponentially) the possible groupings of words would be, resulting in fewer new user-words considered or added to the dictionary. Information models with rich (layered) semantics provide a large **m** for the initial design of user-words, and consequently lead to less ambiguity, fewer possible interpretations, and fewer new user-words. When **m** reaches a reasonable scale, permutations of user-words become potentially unnecessary. Cases do not directly change **m**, but do help resolving ambiguity due to insufficient user-words, and hence help to reduce the need for new user-words. Information models and cases represent a tightly structured, efficient kernel of meaning with which the users will be familiar.

The core logic of the metadata search approach proceeds this way. Given a graph **R = <V, E>** of the reference dictionary where **V** is a set of cases (**C**), user-words (**K**), information models (**M**), and database values (**D**); and **E** is a set of edges connecting them:

Step 1: Identify an ordered k-tuple $\mathbf{I}_{\text{keyword}} = (t_1, t_2, \ldots, t_k)$ where t_i is a word/phrase (keyword) in input that also belongs to **K, M or D** of **R**; $i = 1, 2, \ldots, k$; and k is a number of keywords found in the input. Associated with each element t_i is the set of referred **M or D** (each element of $\mathbf{I}_{\text{keyword}}$ may refer to many elements in **M or D**). Denote this set as V_i. Therefore, we have an ordered k-tuple $\mathbf{V}_{\text{keyword}} = (V_1, V_2, \ldots, V_k)$ where V_i is a set of referred **M or D** for t_i.

Step 2: Determine the most similar past case by matching keywords of the query with problem definitions of past cases. If a perfectly matched case is found, then apply the case solution and go to Step 5. Otherwise if the similarity measure of the most similar case is sufficient significance (say at least 60%), then modify the case to obtain a solution and go to Step 5.

Step 3: Determine a minimal combination set of elements of $V_1, V_2, \ldots, V_k$.

Step 4: Search for the best interpretation by using the branch and bound method.

Step 5: Map the result to the database query language. Obtain the results of query and confirm them with users.

Note that the case-based learning mechanism engages user in dialogue as needed. Its outcome becomes new cases and user-words added to **C** and **K**, respectively. The above logic consists of a few postulates.

Postulate 1: Necessary Condition

The natural query input contains at least one keyword found in **R**.

Postulate 2: Sufficient Condition

The natural query input contains an unambiguous set of keywords necessary for forming an accurate SQL statement for the query.

Postulate 3: Optimality

When all other things are equal, the best interpretation of the natural query input is the one requiring minimum traversal on the logical structure of **R**.

Although the sufficient condition guarantees the logic to succeed, it is not necessary. The metadata search approach actually works under the necessary condition in most cases (see Section 5). Postulate 3 also allows for adding new criteria such as operating rules of the database to enhance the search.

3. Making the Approach Work: the Core Methods and Algorithms

3.1. The Reference Dictionary of Metadata

We use a metadata representation method based on the Metadatabase model [Hsu et. al., 1991] to create the reference dictionary. Previous works have shown the model to be scalable (i.e., metadata independence) and established its feasibility and appropriateness for meeting the requirements of this research [Hsu, et.al., 1991 and Hsu 1996]. The other benefits of using this model include its capability to incorporate rules [Bouziane and Hsu, 1993, 1997; Babin and Hsu 1996] and to support global query processing across multiple databases [Cheung and Hsu, 1996]. A modeling system helps the development and creation of the Metadatabase [Hsu, et.al. 1993; Hsu 1996] using the Two-Stage Entity-Relationship (TSER) modeling Method [Hsu et.al., 1991]. The reference dictionary expands the original representation method (for enterprise information models) to include the additional metadata of database values, user-words, and cases in an integrated (connected) structure for the entire reference dictionary.

The structure of reference dictionary, shown in Figure 1, represents either a table of metadata or a particular type of integrity control rules. The metadata include subjects and views, entity-relationship models, contextual knowledge in the form of rules, application and user definitions, database definitions and database values. User-words are defined as ordered pairs (class, object). Classes include Applications, Subjects, EntRels (entity-relationship), Items, Values, and Operators; all of which are metadata tables as shown in Figure 1. Objects are instances (contents) of these classes. An object is uniquely identified by an ordered quadruple (Item name, EntRel name, Subject name, Application name) as well as an identifier. A case in the case-based reasoning paradigm typically consists of three components: a problem definition, a solution, and its outcome [Kolodner, 1993]. New problems would use the problem definition to find the (best) matching cases and apply associated solutions to them. The third component is useful only when the domain knowledge is incomplete or unpredictable. In this research, however, the reference dictionary contains complete domain knowledge needed, so the problem definition component is expanded but the outcome is dropped. A set of keywords and their recognized vertices for a query describes the problem definition and its interpretation defines the solution.

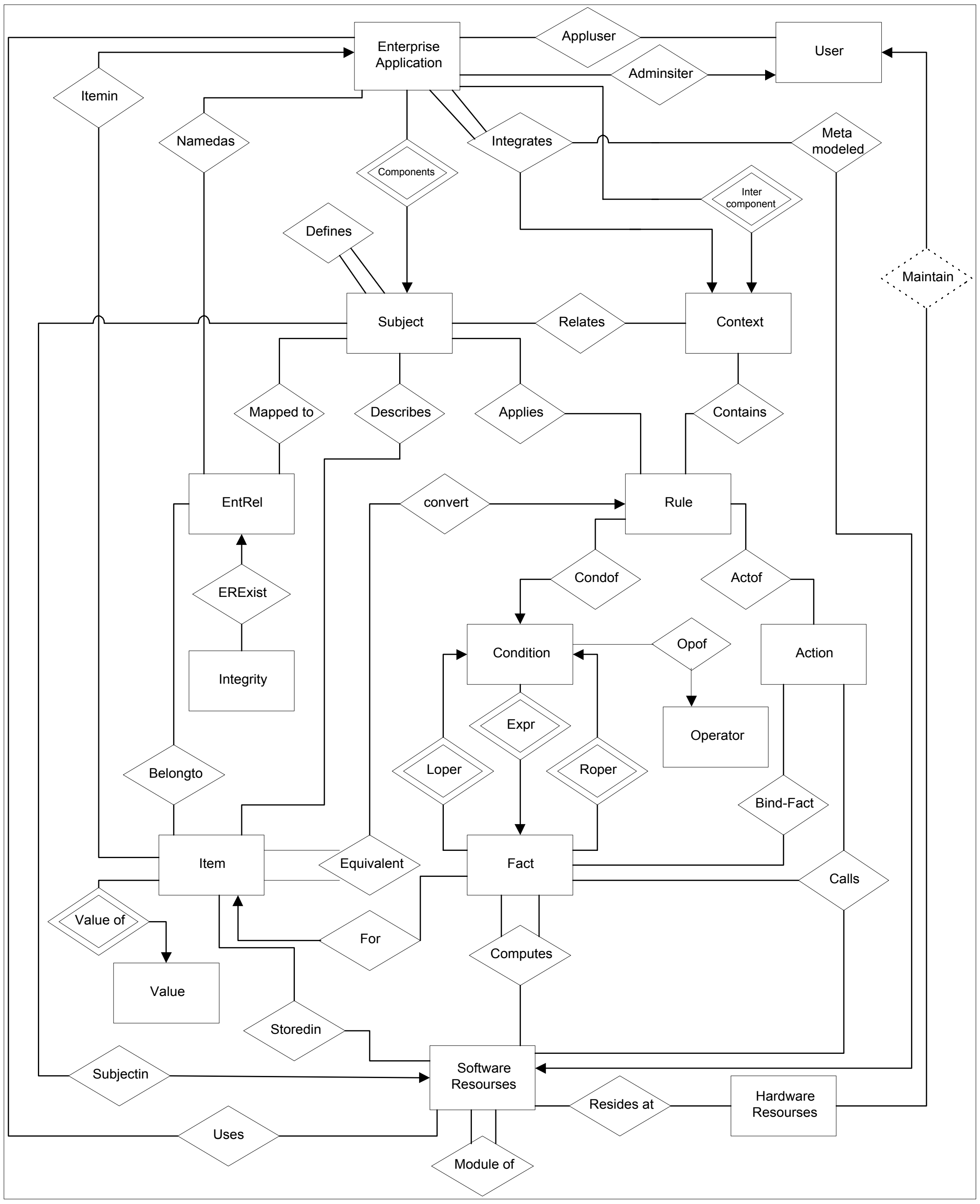

Figure 1: The structure of reference dictionary

3.2. The Graphical Representation of Natural Language Queries

Interpretations of a natural language query are defined on a graph G (Definition 1), a sub-graph of the reference dictionary graph. Given a *natural language query* Q (Definition 2), the natural language interface performs interpretation in several steps. It first scans Q to recognize the keywords (entries in the reference dictionary) in the natural query, i.e., the *recognized keywords* (Definition 3). It then determines their corresponding *recognized vertex sets* (Definition 4) in G and identifies all *query images* (Definition 5) of Q on G based on these vertex sets. Since Q may be ambiguous (e.g., incomplete and multi-valued mapping) to G, each of its recognized keywords could correspond to multiple recognized vertices, resulting in multiple query images. Further, a recognized vertex may not always connect to other recognized vertices in a way covering a complete range of data semantics (database value, attribute, entity, and relationship) with a unique path. Therefore, it could have multiple *semantic paths* (Definition 6). Taking these data semantics into account results in all possible semantic paths for a query image, called *feasible graphs* (Definition 7). The refinement of feasible graphs leads to *connected feasible graphs* (Definition 8) and *complete query graphs* (Definition 9). *A complete query graph represents an executable interpretation of the natural language query Q according to the logical structure G*. The branch and bound algorithm implicitly searches all possible interpretations to determine the final query graph for execution.

Definition 1: A graph G is a graph <V, E>, where sets V and E are defined on the reference dictionary. In particular, V is a set of vertices of five types: subjects, entities, relationships, attributes, and values; and E is a set of their connection constraints (owner-member and peer-peer associations). Owner-member constraints belong to two types: subject-(sub)subject-entity/relationship-attribute-value and subject-attribute. Peer-peer constraints belong to three types: entity-entity, entity-relationship, and relationship-relationship.

Definition 2: A natural language query Q is a string of characters segmented by spaces.

Definition 3: A recognized keyword t_i of Q is a segment of Q matching some entries in the reference dictionary or some vertices in graph G.

Definition 4: A recognized vertex set of a recognized keyword t_i , V_{t_i} , is a set of vertices of G that matches t_i. A member of V_{t_i} is a recognized vertex.

Definition 5: Given an n-recognized-keyword query where $n \neq 0$, a query image in graph G is a set of recognized vertices v_i where $i = 1, \ldots, n$ and $v_i \in V_{t_i}$, respectively.

Definition 6: A semantic path of a recognized vertex v_i is a minimum set of vertices in G containing v_i that satisfies the following conditions: it contains an entity/relationship vertex and its vertices are connected. The vertices it contains, other than v_i, are all implied vertices by v_i to provide an interpretation (semantics) of v_i.

Definition 7: A feasible graph of an n-recognized-vertex query image, where $n \neq 0$, is a collection of semantic paths sp_i where $i = 1, \ldots, n$ and sp_i is a semantic path implied by a recognized vertex v_i of the query image.

Definition 8: A connected feasible graph is a connected sub-graph of G containing the feasible graph and a collection of entity/relationship vertices and edges in this graph. This collection is minimally sufficient to connect entity/relationship vertices of the feasible graph. The path connecting these entity/relationship vertices is called entity/relationship solution path.

Definition 9: A query graph is a subgraph of connected feasible graph. It consists of <V, E> where V is a set of entity/relationship vertices, attribute vertices, and value vertices and E is a set of edges connecting them.

Consider a CIM database encompassing three application systems: order processing, process planning, and shop floor control. Figures 2a and 2b show the sub-graph for order processing system. In Figure 2a, the path "S ORDER_PROCESSING - S ORDER - E PART – I opsI_100 part_id – V PZ1" shows a "Subject-(sub)Subject-Entity/relationship-Item/attribute-Value" hierarchy, and the "E PART - E ORDER_ITEM - E ORDER - E CUSTOMER" path shows a peer-to-peer constraint for Entities. The paths in Figure 2b show the subject-attribute hierarchical constraint for this sub-graph.

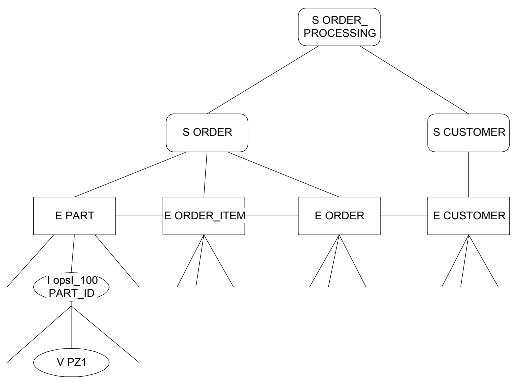

Figure 2a: A View of the Sub-Graph for Order Processing System

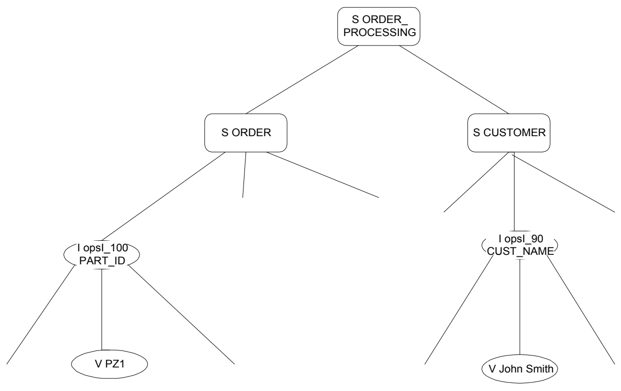

Figure 2b: Another View of the Sub-Graph for Order Processing System

To interpret this Query 1: “Which customers placed orders on PZ1? Just give me their names,” Table 1 illustrates the recognized keywords and their recognized vertices.

Keyword	Vertex Set
CUSTOMERS	{ E CUSTOMER }
ORDERS	{ S ORDER }
PZ1	{ V opsl_100|PZ1, V ppsl_54|PZ1, V sfcl_11|PZ1, V sfcl_5|PZ1 }
NAMES	{I opsl_90}

Table 1. Recognized Keywords Vertex Sets for Query 1.

The following are possible query images generated from Table 1:

QI 1: {E CUSTOMER, S ORDER, V opsI_100|PZ1, I opsI_90}

QI 2: {E CUSTOMER, S ORDER, V ppsI_54|PZ1, I opsI_90}

QI 3: {E CUSTOMER, S ORDER, V sfcI_11|PZ1, I opsI_90}

QI 4: {E CUSTOMER, S ORDER,, V sfcI_5|PZ1 }, I opsI_90} .

Take the first query image QI1, for example. Table 2 shows its semantic paths for recognized vertices. Note that there are two semantic paths for the recognized vertex V opsI_100|PZ1. A semantic domain for a semantic path spans all members of the semantic domain. For example, the semantic domain for the semantic path (E CUSTOMER) spans all attributes belonging to this entity and all values belonging to these attributes.

Recognized Vertex	Semantic Path
E CUSTOMER	{ (E CUSTOMER) }
S ORDER	{ (S ORDER, E PART), (S ORDER, ORDER_ITEM), (S ORDER, E ORDER)}

V opsI_100|PZ1	{ (VPZ1, I opsI_100, E PART) }
	{(VPZ1, I opsI_100, E ORDER_ITEM) }

NAMES	{(I opsI_90, E CUSTOMER)}

Table 2. Recognized Vertices and Their Semantic Paths.

Based on the semantic paths in Table 2, two feasible graphs (FG11 and FG12) for QI1 are determined (Figures 3 and 4, respectively). The connected feasible graph CFG111 and its query graph QG111 for the feasible graph FG11 are determined as shown in Figures 5 and 6, respectively.

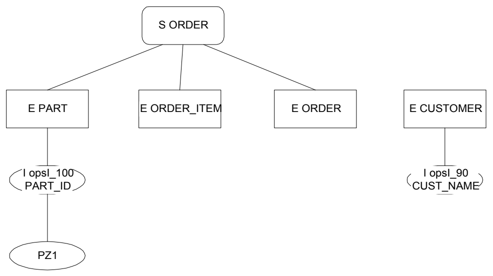

Figure 3: The feasible graph FG11 for QI1.

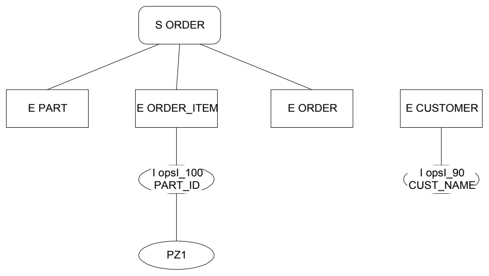

Figure 4: The feasible graph FG12 for QI1.

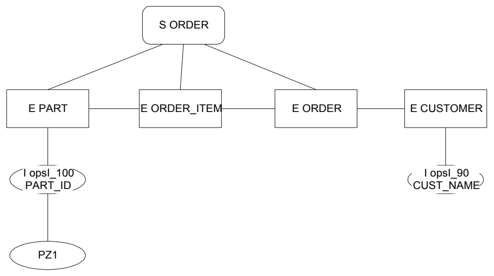

Figure 5: The connected feasible graph CFG111 for FG11.

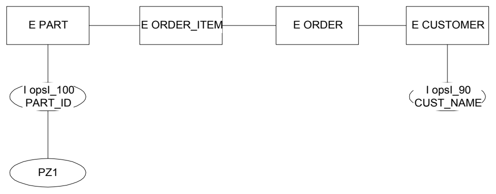

Figure 6: The query graph QG111 of CFG111.

3.3. The Interpretation Algorithm

The object of the search algorithm is to determine the best query graph (interpretation) among all possible query graphs for a natural language query. We use Postulate 3 (see Section 2) to develop the objective function for the search: the cost (z) of a query graph, measured by the count of its edges. Since a query graph is a tree, its cost is |V| - 1 where V is the vertex set of the query graph. We use the same procedure discussed above to complete the development of all possible interpretations for Query 1, along with their cost. Figure 7 shows the complete results. The best query graphs for this query is a query graph with cost z = 6.

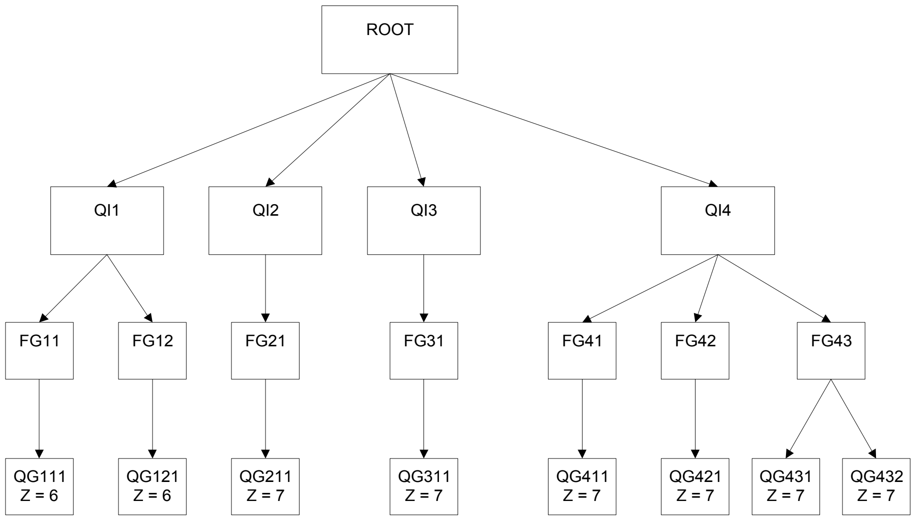

Figure 7: The Query Graph Enumeration Procedure for Query 1.

Note that Figure 7 shows multiple minimum-cost query graphs (interpretations) for Query 1. These minimum-cost query graphs could be equivalent. A query graph a is equivalent to a query graph b if any of the following conditions satisfy:

1. Entity/relationship solution path of query graph a is equal to entity/relationship solution path of query graph b;
2. A set of attribute vertices of query graph a is equivalent to a set of attribute vertices of query graph b; or

3. A set of value vertices of query graph a is equivalent to a set of value vertices of query graph b.

In condition 2, two attribute sets are equivalent if every element of one set has an equivalent attribute in the other; and two attributes are equivalent if they have the same attribute vertex belonging to the same entity/relationship vertex. In condition 3, two value sets are equivalent if every element of one set has an equivalent element in the other; and two values are equivalent if they have the same value vertex, attribute vertex, and entity/relationship vertex. For Query 1, QG111 and QG121 are equivalent - see Figure 8.

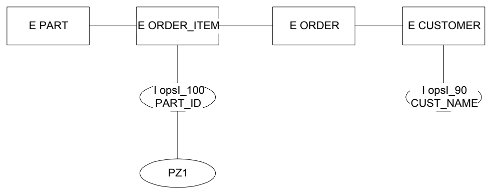

Figure 8: The Query Graph QG121 for FG12.

The search algorithm itself follows the Branch-and-Bound logic. Given an input I = (w_1, w_2, …, w_n), where n is the number of words in a natural language query, the search proceeds as follows:

Step 1: Determine recognized keywords and their recognized vertex sets.

Step 1.1: Determine a set of recognized keywords in the input (we denote this set as $I_{keyword}$) such that these keywords are elements of or indirect references to elements of the graph G. Therefore, we have $I_{keyword}$=(t_1, t_2, …, t_k) where k is the number of keywords formed from n words of I and $k \leq n$.

Step 1.2: Determine an ordered k-tuple $V_{keyword}$=(V_1, V_2, …, V_k) such that V_i is a set of recognized vertices of its correspondent $t_i \in I_{keyword}$ where i=1, 2, ..,k.

Step 2: Determine a minimal set of query images.

Step 2.1: For each $V_i \in V_{keyword}$ where i=1, 2, ..,k and $|V_i| \geq 2$, determine $minV_i$ such that there does not exist an entity/relationship or subject vertex $\in minV_i$

belonging to a subject vertex $\in$ $minV_i$. Therefore, we have an ordered k-tuple V_{min}=($minV_1$, $minV_2$, …, $minV_k$).

Step 2.2: Determine $V_{minVertexSets}$ of V_{min} such that it contains no $minV_i$ that $|minV_i|=1$ and its element is an entity/relationship or subject vertex belonging to $minV_j$ where $|minV_j| = 1$, $i \neq j$, and j=1,2, …, k. Therefore, we have $V_{minVertexSets}$=($minV_1$, $minV_2$, …, $minV_{k*}$) where $k^* \leq k$. Corresponding to $V_{minVertexSets}$ is $I_{minKeywords}$=(t_1, t_2, …, t_{k*}).

Step 2.3: Determine a query image set QI such that QI = {QI_i | QI_i is a set of k* recognized vertices, {$v_{1,i} \in minV_1$, $v_{2,i} \in minV_2$, …, $v_{k*,i} \in minV_{k*}$}; i = 1, …, n ; n = $|minV_1||minV_2|\ldots|minV_{k*}|$ where $|minV_j|$, j = 1, 2, …,k*}.

Step 2.4: Determine a minimal set $minQI_i$ for each $QI_i \in QI$ such that $minQI_i$ = { v_j | for all entity/relationship or subject vertex v_j , there does not exist subject vertex v_l such that $v_j \in v_l$ where $j \neq l$; j, l = 1, …, k**; $k^{**} \leq k^*$}.

Step 2.5: Determine a minimal QI set, QI_{min} = { $minQI_i$ | for all $minQI_i$, there does not exist $minQI_j$ such that $minQI_I$=$minQI_j$ where $i \neq j$; i, j = 1, …, n*; $n^* \leq n$}.

Step 3: Search for the best interpretation (query graph).

Search for the query graph QG_k such that cost(QG_k) = min(cost(QG_1), cost(QG_2), …, cost(QG_m)) where QG_1, QG_2, …, QG_m are all possible query graphs enumerated from QI_{min} and cost(QG_i) = (number of vertices of QG_i) – 1 where i = 1, 2, …, m. Since there could be multiple query graphs satisfying this condition, we have QG_{opt} = { QG_k | cost(QG_k) = min(cost(QG_1), cost(QG_2), …, cost(QG_m)) and k $\in$ {1, 2, …, m}}.

Step 4: Remove equivalent query graphs in QG_{opt}.

If $|QG_{opt}| > 1$, then determine QG_{case} from QG_{opt} such that for all $QG_i \in QG_{case}$ there does not exist $QG_j \in QG_{case}$ that (1) entity/relationship solution path of QG_i is equal to entity/relationship solution path of QG_j, (2) set of attribute vertices of QG_i is equivalent to set of attribute vertices of QG_j, and (3) set of value vertices of QG_i is equivalent to set of value vertices of QG_j.

Step 1 aims at text processing and is self-evident. Step 2.1 removes redundant recognized vertices for each keyword using owner-member relationship subject-

(sub)subject-entity/relationship. This step maximizes semantic domains and minimizes possible query images. Step 2.2 employs similar idea to remove redundancy between unambiguous keywords. When all recognized vertices for a keyword are removed, the keyword is removed too. Step 2.3 enumerates query images from sets of minimal recognized vertices obtained from Step 2.2. Step 2.4 removes redundant elements in each query images based on owner-member constraints (i.e., subject-(sub)subject-entity/relationship. Step 2.5 removes identical query images to minimize the number of enumerated query graphs.

Overall, Step 2 is important for metadata search when the enterprise information models include layers of semantics (subject hierarchy in this model or object hierarchy in general) on top of database structures. Layered semantics helps resolving ambiguity (e.g., completing the query), but also leads to redundant interpretations. The semantic model employed for the CIM example is complex and difficult to show in paper. Therefore, for the purpose of clear illustration, we slightly modified the actual results that the algorithm generates for Query 1, as shown in Table 1. However, the basic logic shown in Section III.B remains accurate and the tables and figures there still represent a close illustration of how the algorithm works. In the actual results, Customer Subject replaces the Customer Entity, which the subject owns, and drives the ensuing derivation (see [Boonjing 2002] for detailed documentation of this actual process).

The search problem in Step 3 is an optimization problem with objective function z(t) where t is a terminal vertex (a query graph). The problem is to minimize z(t) with respect to graph G. An evaluation function LB(v) finds the lower bound for an intermediate vertex v (a query image or a feasible graph), so that the search could either fathom all paths starting with v or pick the most promising v to explore further. This lower bound of a vertex indicates the best minimum-cost query graphs that could be obtained if the vertex is picked to explore further.

A lower bound of a query image is calculated based on (1) the number of its value vertices, (2) the number of its attribute vertices, and (3) the number of its entity/relationship implied vertices (entity/relationship vertices implied from recognized vertices). The calculation is under the condition that only entity/relationship implied vertices are included in an entity/relationship solution path (i.e., all entity/relationship implied vertices are connected). It first creates a base set of entity/relationship vertices – the base set includes all entity/relationship vertices and all entity/relationship vertices implied from subjects. It then sets the number of counted vertices to 0. For each attribute or value vertex, it increases the number of counted vertices by 1 and determines all entity/relationship vertices it belongs to. If there does not exist any of these entity/relationship vertices in the counted entity/relationship set, then increase number of counted vertices by 1 (i.e., implied adding a new entity/relationship vertex to the base set). Otherwise if there does not exist any of these entity/relationship vertices in the base set, increase the number of counted vertices by 1 and add them to the counted entity/relationship set. Thus,

- LB(v) = |base set (query image)| + the number of counted vertices –1 if v is a query image; or
- LB(v) = (total number of its value, attribute, and entity/relationship vertices) -1 if v is a feasible graph.

With a standard branch-and-bound algorithm [Kumar, 1987; Nau et. al., 1984] and the evaluation function LB(), the implicit search graph for the Query 1 is shown in Figure 9.

As described earlier, query graph QG111 is equivalent to QG121. Therefore, one of them can be removed - suppose it to be QG121. Then, QG111 is the final interpretation of the Query 1.

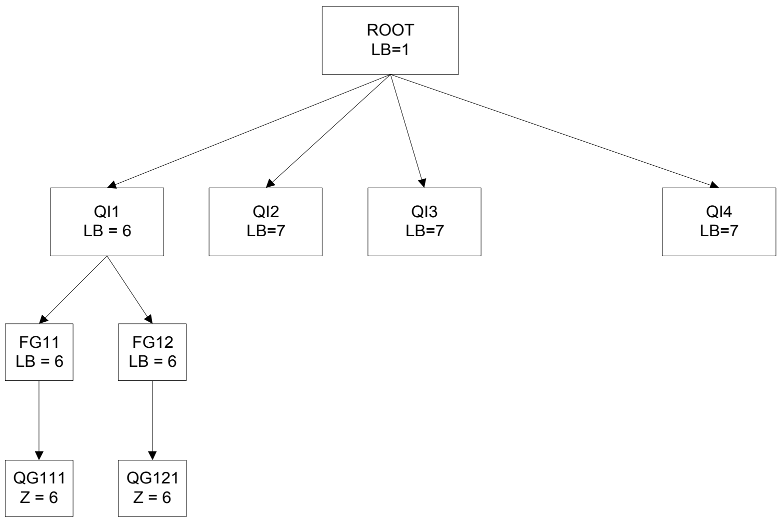

Figure 9: The Implicit Search Graph for Query 1.

The above algorithm is a backbone to the interpretation, but it is also a bare bone version. Enhancements such as better lower bounds and additional constraints based on business rules and other contextual knowledge of the enterprise databases could improve the performance of the algorithm and its power to resolve ambiguity.

3.4. Database Query Language Generation

The next step is to map the final query graph into the target database query processing language. This research has developed the query generator for the Structured Query Language (SQL). The query generator infers from the query graph the entities and relationships and their join conditions that the query involves. It then determines the database objects and values required, and put these operators and parameters into an SQL statement.

The SELECT list is determined by rules based on appearance of attribute vertices in a query graph as follows.

- If the query graph contains attribute vertices, the select list contains those attribute vertices and all attributes of value vertices.
- If the query graph does not contain any attribute vertices, the select list contains all attributes that belong to entity/relationship vertices and all attributes of value vertices.

The selection condition is determined by rules based on appearance of value vertices in a query graph as follows.

- If v_1 belong to attribute A_1 of entity/relationship R_1; v_2 belong to attribute A_2 of entity/relationship R_2; …; and v_n belong to attribute A_n of entity/relationship R_n, then the selection condition is "($R_1.A_1 = v_1$ AND $R_2.A_2 = v_2$ AND … AND $R_n.A_n = v_n$)."
- If v_1, v_2, …, v_n belong to attribute A of entity/relationship R , then selection condition is "(R.A = v_1 OR R.A = v_2 OR … OR R.A = v_n)."

For example, query graph QG111 leads to the following information:

Target attributes: CUSTOMER.CUST_NAME, PART.PART_ID
ER list: ORDER_ITEM, PART, ORDER_HEADER, CUSTOMER
Join Condition: ORDER_ITEM.PART_ID=PART.PART_ID and ORDER_ITEM.CUST_ORDER_ID=ORDER_HEADER.CUST_ORDER_ID and ORDER_HEADER.CUST_ID=CUSTOMER.CUST_ID
Selection Condition: (PART.PART_ID = 'PZ1')

Hence, the corresponding SQL statement for the original Query 1 is:

```
SELECT DISTINCT CUSTOMER.CUST_NAME, PART.PART_ID
FROM ORDER_ITEM, PART, ORDER_HEADER, CUSTOMER
WHERE ORDER_ITEM.PART_ID=PART.PART_ID and
ORDER_ITEM.CUST_ORDER_ID=ORDER_HEADER.CUST_ORDER_ID and
ORDER_HEADER.CUST_ID=CUSTOMER.CUST_ID and (PART.PART_ID = 'PZ1')
```

3.5. Case-Based Reasoning and Interaction

In theory, the system could accumulate all natural queries it has processed (using the above method) into a case-base. When the case-base has grown to a sufficient size, the designer would have a choice to shift to relying mainly on case-based reasoning to answer new queries and uses the interpretation algorithm only as a backup. This design would be reasonable since the growth of cases is linear. However, in the paper, we still regard cases as a secondary tool to support the primary means of metadata search. As such, the role of case-based reasoning is to assure a closure to the user and in a sense a learning capability to the system. Therefore, we consider in this paper only the need to build new cases for natural queries that the system failed to provide a single (query graph) answer or the user rejected the answer. An interaction with the user is the means to achieve this end. The interaction capability is more a last defense than a regular way of solution; however, it could improve the performance in the worse than average cases when the repository of the cases grows. When an interaction becomes necessary, the system asks the user for additional information to solve the query correctly. There are three possible causes of failure: (1) incorrect recognized vertex for a recognized keyword, (2) no recognizable keywords in a query, and (3) insufficient keywords. A metadata-based interaction identifies the cause and guides the user to define new user-words to fix it. For the first cause, it provides the pertinent metadata (information model) describing the possible recognized vertices for the keywords found so that the user could either designate the intended vertex for a keyword or redefine the keywords. For the second cause, it walks the user through the enterprise information model from high level down to associate new user-words to vertices of information models. For the last cause, it similarly guides the user to provide additional keywords for the query. In all these cases, the interaction adds new user-words as well as building cases to the reference dictionary. The system will repeat this cycle until it gives a correct result. Another design to resolve the case of multiple query graphs for a query is to present all possible solutions to users in terms of natural language for them to choose the intended solution. If the number of alternative solutions is relatively large, it presents all possible scopes of solutions for users to choose to narrow down the possible solutions before asking them to choose the intended solution. When the system obtains the correct result, it completes the case with problem definition, solution definition, and other information according to the structure of cases in the Reference dictionary.

The method of case-based reasoning uses the cases built and accumulated either to assist the metadata search or to answer directly the queries. The method includes (1) a matching mechanism to match a query with a case, (2) criteria to decide whether to reuse the most similar case, and (3) a mechanism to reuse the case. A standard result in the information retrieval field to match a query with a case is based on the vector space model [Salton and Mcgill, 1983] and its COSINE measure of similarity. A procedure for the COSINE measure compares a set of keywords representing a query (Q) and a set of keywords representing a case (C), as follows.

```
Procedure cosine_simialarity (Q, C) {
    Determine a set of common keywords of Q and C.
    For each keyword in C that do not belong to the set of common keywords
    {
            If (its recognized vertex set matches with the recognized
            vertex set of keyword in Q that is not an element of the set
            of common keywords) {
            Substitute the keyword with the matched recognized-vertex
            keyword of Q.
        }
    }
    Let base_set= $Q \cup C$
    Let U=binary_vector(Q)
    //function binary_vector (Q) returns a binary vector based on base_set
    Let V=binary_vector(C)
    Return (U.V)/(|U|)(|V|)
}
```

Since keywords in Q could be synonyms of keywords in C, the determination of synonyms is included in this procedure to assure preciseness of the COSINE measure. Keywords of retrieved C are substituted by their synonyms in Q prior to determination of the COSINE similarity. A given query is matched to all cases based on this procedure to identify the most similar case. When the perfect match (the COSINE measure is 1) is found for the query, the case solution will be a solution for the query. But if the COSINE measure is between 0 and 1, the method must decide at which level it will reuse the case. The degree of match is also a basis determining how the system reuses the case. It could reuse the case without modification, or engage the user in a metadata-based interaction to modify the solution definition. Finally, alternative measures such as neural networks and simulated annals are possible alternatives to the COSINE measure for the case-based reasoning method to adopt.

4. Implementing the Approach

We implemented the core algorithms in a software system running on UNIX-Oracle Server. The reference dictionary and the application database - a Computer-Integrated Manufacturing (CIM) system - both reside in Oracle. The software itself has two components. The user interface is coded in HTML and Java Script to be compatible with major Web browsers such as Internet Explorer and Netscape. The core algorithms use Java Applets embedded in the user interface in a browser-based computing manner.

Therefore, the software is portable to multiple platforms and databases, and supports use via the Internet.

The implementation developed so far solves primarily natural language queries that have a basic SQL solution. The basic SQL uses search-conditions in the WHERE clause that either (1) are an expression of the form "attribute_1 = attribute_2" or "attribute = value," or (2) a combination of such expressions with the logical operators "AND" and "OR." These results are sufficient to show the feasibility of the metadata search as we will discuss in the next section. Nonetheless, they are insufficient for implementing the approach in practice. This section describes possible extensions to make the metadata search approach practical. We wish to point out that these extensions are still based on the basic methods and require only time and effort to develop. The conceptual model discussed in Section II covers the logic of these extensions.

Foremost, the algorithms should extend to include the full range of SQL capability, especially the other standard operators not included yet, derived-attributes and expressions, and date-related and other temporal selection criteria. This extension can result from a straightforward inclusion of additional database operators and user-words into the reference dictionary. That is, all these extended SQL capabilities use finite and SQL-defined terms and phrases, which are exhaustively recognizable for incorporation into the reference dictionary like other database operators; and so do their user-words. Relatively speaking, developing user-words for temporal criteria could be more involved than for other database operators. However, temporal expressions in practice are reasonably finite since most people use only clear-cut ways to express time in their requests against definitive applications. We, therefore, can develop particular user-words to capture these possibilities with finite efforts. When necessary, we could also develop special rules to complement user words for some of the database operators and expressions. These rules would recognize certain specific patterns of natural wording corresponding to the few select keywords, to reduce the reliance on user-words. Operators for comparison, aggregation, ordering, and grouping are some of the candidates for which the system can offer optional special functions to repeated users (e.g., display them on screen in a side bar). This method can also extend to the handling of derived attributes (arithmetic expressions defined on set of attributes and constants).

We expect the users of these advanced features to be direct database users, who are familiar with such SQL concepts.

In addition, the system should furnish a set of productivity tools to help particular enterprises implement the metadata search approach and maintain the reference dictionary. They should include (1) information modeling tool, (2) user-word acquisition tool, (3) database value acquisition tool, and (4) user dialog tool. The information modeling tool assists the designers to develop enterprise information models and populate them into the reference dictionary. The user-word acquisition tool helps the designers to browse enterprise users through the information models and obtain promising user-words, primarily at the development time but also applicable to training new users during run time. The database value acquisition tool links automatically the run time database values of the enterprise database to the reference dictionary. The user dialog tool facilitates the design of the metadata-based interaction discussed above.

5. Verifying the Approach in a Laboratory Testing

We tested the system with 10 users in an attempt to substantiate the claims of the metadata search approach. The participants included professionals not associated with Rensselaer as well as undergraduate students and graduate students at Rensselaer. Their background in databases ranges from strong to virtually none. We provided the participants with the descriptions, information models, and contents of the CIM database; but were otherwise completely detached so as to allow them to ask as many questions as they wished against the database in any text forms. An answer to a question is correct when it includes all information the user required. The reference dictionary of the testing case included only information models, database values, and less than 200 user-words that we created by ourselves in advance. (See [Boonjing 2002] for a complete documentation of the implementation and testing].)

Analysis I: Truly Natural Query Processing

Result: the software solved every query listed in Section I.

The list of queries includes different forms of text inputs, including sentences, short essays, and phrases; and some of them are even incorrect grammatically. They do not follow any discernible patterns, and many of them do not even contain sufficient information to allow query formulation using SQL or any other fixed syntax. This result supports the claim that the metadata search approach is capable of processing truly natural language queries, including "incomplete queries."

Analysis II: Scalability and Soundness

Result: the software used a limited number of new user-words to solve every query listed below. The software failed originally to produce correct answers to these queries; however, it solved every one when the boldface words/phrases that the users articulated were added to the reference dictionary as user-words (only one for each). For example, the software delivered a single correct solution to the query “William Batt’s **contact**” when the word “contact” was added to the reference dictionary as a new user-word associated with the attribute “B_ADDR” (billing address). Without it, the system produced multiple interpretations for user to select.

- William Batt's **contact**
- Michael Goin **address**
- Not done yet **transaction**
- **Buyers** of PZ10.
- How many **people** order pz1?
- Is Jim Kowlaski billing address and **mailing address** the same?
- What is Craig Norman billing address and **home address**? Please also list all of his orders.
- What are the **products** associated with order no 00001?
- Show the detail Craig Norman’s order, part description, quantity, and **where to ship** it?
- Find the information [order_id, customer_id, desire_date] of order, which is **not completed**.
- List all **in-process** orders and their details.

The important fact about these new user-words is their consistency with the basic logic of metadata search: they do not require n-gram and their number grows linearly. The result shows, the concept of metadata search is sound and the approach performs as designed. It also supports the claim that enterprise information models with layered semantics reduce the number of user-words required and enhance the metadata search.

Analysis III: the Necessary Condition and the Sufficient Condition

Result: all queries tested satisfy the necessary condition and the sufficient condition postulated in Section II.

Most of the actual queries tested contain less information than what SQL requires, and none more. Thus, the sufficient condition seems evident. We now go beyond these actual testing cases to examine the necessary condition. The basic question here is, how well the metadata search approach handles ambiguity due to incomplete queries. Some of the queries solved contain only one keyword; but how general is this result? Consider the query “How about part PZ1?” which consists of two recognized keywords: “part,” and “PZ1.” With these two keywords, the software generated the correct answer. If we removed the keyword “part” and asked only “How about PZ1?,” then the system would face an ambiguity that it cannot resolve because “PZ1” in the CIM database corresponds to multiple database objects. Consequently, the software would generate seven possible interpretations either for the user to choose via an interaction mechanism, or for resolution using the case-based reasoning. In either case, a definitive closure is still achievable since the above alternatives do in fact contain the single correct answer to the user’s question. This example shows that a natural query is solvable if it contains at least one keyword. Based on this observation, we assert the fundamental correctness of the metadata search approach.

Analysis IV: Response Time

Result: the response times of all queries were between 1 and 10 seconds, running the software from a Web client site against the Oracle server.

The response time recorded is the total time for all steps involved in the Web query. Although they seem reasonable, we recognize the need to improve the performance further; and case-based reasoning is a promised method to this end. As discussed in Section III.E, a case retains the keywords and their recognized vertices for a query as the problem definition and its final query graph as the solution. Thus, the system could reuse the solution for a new query matched to the case without going through the search graph. For example, the branch-and-bound search generates 128 query images, 16 feasible graphs, and 8 query graphs for the query "Part pz1, pz2, ba, and bb…" shown in Section I. A matching case could reduce the response time from almost 10 seconds to nearly 0 seconds.

6. Contemplating the Future

We establish in the paper a metadata search approach as a feasible solution to the problem of natural language database query. The new approach features a new class of reference dictionary integrating four classes of enterprise metadata: enterprise information models, database values, user-words, and query cases. The reference dictionary accommodates all possible interpretations of natural language query pertaining to enterprise databases and promises to reduce the growth of user-words through enterprise information models. The branch-and-bound search method makes it possible and efficient to search all possible (machine) interpretations of a natural language query and to determine the optimal solution. The approach also provides interactive capability and case-based reasoning to assure successful closure to a query and to improve performance. A number of alternative designs of the collaboration between query cases and search algorithms are permissible.

Testing results of a limited implementation of the core method in a laboratory setting suggest that the metadata search approach is fundamentally sound and correct, as claimed in Section II. The core system is evidently capable of processing truly natural language inputs in text form, including short essays, sentences, and phrases and words, under certain conditions. The necessary condition is the text input contains at least one recognized keyword (an entry found in the reference dictionary). The sufficient condition is the text input contains a complete set of information that a single SQL answer requires. The testing results also confirm that the performance seems reasonable (1 - 10 seconds), but case-based reasoning would help.

These results suggest a worthy continuation of the effort. Future research opportunities exist in a number of fronts. The first area is the necessary tools and enhancements to make the new approach practical. They include the design of the metadata-based interaction, case-based reasoning, and stronger search criteria for the interpretation algorithm. They also include productivity tools to facilitate the acquisition of user-words and other metadata at development time. Furthermore, the new reference dictionary and its graphical representation are amenable to developing truly natural language capability for object-oriented databases as well as for multi-databases. Voice capability will be critical for lay persons to become direct users of databases. Thus, linking text input and output to voice technology and making necessary adjustments to the interaction methods promise to be worthy extensions to the approach. Finally, many general applications of natural language interface involve information retrieval of some kind. To the extent that metadata make sense to these applications, the metadata search approach should have some promises for their natural language interface. Research into this general area beyond databases per se is a possibility.

References

Adam, N.R. and A. Gangopadhyay, "A Form-Based Natural Language Front-End to a CIM database," *IEEE Trans. On Knowledge and Data Engineering,* 9:2, 1997, pp. 238-250.

Babin, G. and C. Hsu, "Decomposition of Knowledge for Concurrent Processing," *IEEE Trans. Knowledge and Data Engineering*, 8:5, October 1996, pp. 758-772.

Bates, M., M.G.Moser, and D. Stallard, "The IRUS Transportation Natural Language Interface," in *Expert Database Systems,* 1986, pp. 617-630.

Bergman, L., J. Shoudt, V. Castelli, and C. Li, "An Interface Framework for Digital Library," *International Journal on Digital Libraries,* 2:2-3, 1999, pp. 178-189.

Boonjing, V., *A Metadata Search Approach to Natural Language Database Query,* Ph.D. Dissertation, 2002, Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY, 2002.

Boonjing, V. and C. Hsu, "Metadata Search: A New Approach to Natural Language Database Interfaces," *Proceedings IASTED International Conference on Information Systems and Databases*, October, 2002, Tsukuta, Japan

Boonjing, V. and C. Hsu, "Natural Language Interaction Using a Scalable Reference Dictionary," *Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems,* Burg, Germany, June 2003.

Bouziane, M. and C. Hsu, "A Rulebase Management System Using Conceptual Modeling," *J. Artificial Intelligence Tools*, 6:1, March 1997, pp. 37-61.

Bouziane, M. and C. Hsu, "A Rulebase Model for Data and Knowledge Integration in Multiple Systems Environments," *J. Artificial Intelligence Tools*, 2:4, December 1993, pp. 485-509.

Brajnik, G., G. Guida, and C. Tasso, "An Expert Interface for Effective Man-Machine Interaction.," in *Cooperative Interfaces to Information Systems*, 1986, pp. 259-308.

Carbonell, J.G., "Learning by Analogy: Formulating and Generalizing Plans from Past Experience," in Machine Learning. 1983, pp. 137-161.

Cheung, W. and C. Hsu, "The Model-Assisted Global Query System for Multiple Databases in Distributed Enterprises," *ACM Trans. Information Systems*, 14:4, October 1996, pp. 421-470.

Codd, E.F, R.S.Arnold, J-M. Cadiou, C.L. Chang, and N. Roussopoulos, "RENDEZVOUS Vesion 1: An Experimental English Language Query Formulation System for Casual Users of Relational Data Bases," in IBM Reaearch Report RJ2144, 1978.

Codd, E.F., "How about Recently? (English Dialog with Relational Databases Using Rendezvous Vesion 1). In B. Shneiderman (Eds). Databases: Improving Usability and Responsiveness, 1978, pp. 3-28.

Fellbaum, C. (Eds), "WordNet: an Electronic Lexical Database," MIT Press, Boston, 1998.

Gross, B.J., D.E. Appelt, P.A. Martin and F.C.N. Pereira, "TEAM: an Experiment in Design of Transportable Natural-Language Interfaces," *ACM Trans*actions, 32, 1987, pp. 173-243.

Guida, G. and Tasso C., "NLI: A Robust Interface for Natural Language Person-Machine Communication," *Int. J. Man-Machine Studies*, 17, 1982, pp.417-433.

Hendrix, G.G. Sacerdoti, E.D. Sagalowicz, C. and Slocum, J., "Development a Natural Language Interface to Complex Data," *ACM Trans. on Database Systems*, 3:2, 1978, pp. 105-147.

Hsu, C., M. Bouziane, L. Ratter, and L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach," *IEEE Trans. on Software Engineering*, 17:6, 1991, pp. 604-625.

Hsu, C., Y. Tao, M. Bouziane, and G. Babin, "Paradigm Translations in Integrating Manufacturing Information Using a Meta-Model: the TSER Approach," *J. Information Systems Engineering*, 1:1, 1993, pp. 325-352.

Hsu, C., *Enterprise Integration and Modeling: the Metadatabase Approach*, Kluwer Academic Publishers, Boston, 1996.

Janus, J.M., "The Semantics-based Natural Language Interface to Relational Databases," in L. Bolc and M. Jarke (Eds), *Cooperative Interfaces to Information Systems*, pp. 143-187, Springer-Verlag, New York, 1986.

Jarvelin, K., J. Kekalainen and T. Niemi, “Concept-Based Query Expansion and Construction,” *Information Retrieval,* 4:3/4, 2001, pp. 231-255.

Johnson, J.A.,“Semantic Relatedness,” *Computers Math. Applic,* 29:5, 1995, pp. 51-63.

Kaplan, S.J., "Designing a Portable Natural Language Database Query System," *ACM Trans. on Database Systems*, 9:1, 1984, pp. 1-19.

Kim, W., “Corpus-Based Statistical Screening for Phrase Identification,” *Journal of the American Medical Information Association,* 7:5, 2000, pp. 499-511.

Kolodner, J.L, “Case-Based Reasoning,” Morgan Kaufmann Publishers, California, 1993.

Maier, D., “The Theory of Relational Databases,” Computer Science Press, Maryland, 1983.

Maier, D. and Ullman, J.D., “Maximal Objects and the Semantics of Universal Relation Databases,” *ACM Trans. on Database Systems*, 8:1, 1984, pp. 1-14.

Meng, F. and W. Chu, "Database Query Formulation from Natural Language using Semantic Modeling and Statistical Keyword Meaning Disambiguation," Technical Report CSD-TR 99003, 1999, Computer Science Department, University of California, Los Angeles, California.

Metais, E., “Enhancing Information Systems Management with Natural Language Processing Techniques,” *Data and Knowledge Engineering,* 41:2-3, 2002, pp. 247-272.

Miller, P.L., "An Adaptive Natural Language System that Listens, Asks, and Learns," in *Advance Papers of Forth International Joint Conference on Artificial Intelligence*. pp. 406-413, Morgan Kaufmann, California, 1975.

Mittendorfer, M. and W. Winiwarter, “Exploiting Syntactic Abalisys of Queries for Information Retrieval,” *Data and Knowledge Engineering,* 42:3, 2002, pp. 315-325.

Mooney, R., *Symbolic Learning for Natural language Processing: Integrating Information Extraction and Querying,* NSF Award Abstract #9704943, 1997-2001.

Motro, A., "Constructing Queries from Tokens," in *Proc. ACM-SIGMOD Int. Conf. Management Data*, pp. 120-131, ACM, Washington D.C., 1986.

Motro, A., “FLEX: A Tolerant and Cooperative User Interface to Databases,” *IEEE Transactions on Knowledge and Data Engineering*, 2:2, 1990, pp. 231-246.

Mylopoulos, J. Borgida, A. Cohen, P. Roussopoulos, N. Tsotsos, Jr. and H. Wong, "TORUS: A Step Towards Bridging the Gap between Data Bases and the Casual User," *Information Systems*, 2:1, 1976, pp.49-64.

Norcio, A.F. and J. Stanley, "Adaptive Human-Computer Interfaces: A literature Survey and Perspective," *IEEE Trans. On Systems, Man, and Cybernetics*, 19:2, 1989, pp. 399-408.

Owei, V., “Natural Language Querying of Databases: an Information Extraction Approach in the Conceptual Query Language,” *Int. J. Man-Machine Studies*, 53, 2000, pp.439-492.

Richardson, S.D., W.B. Dolan and L. Vanderwende, “MindNet: Acquiring and Structuring Semantic Information from Text,” In *ACL’98 CONF 17*, 2, 1998, pp. 1098-1102.

Rissland, E.L. 1984, "Ingredients of Intelligent User Interfaces," *Int. J. Man-Machine Studies*, 21, 1984, pp. 377-388.

Rosenfeld, R., X. Zhu, S. Shriver, A. Toth, K. Lenzo and A. W. Black, “Towards a Universal Speech Interface,” In *Proc. ICSLP 2000,* 2000.

Ruber, P., “Asking Naturally,” *Knowledge Management,* 4:8, 2001, pp. 62-63.

Salton, G. and Mcgill, M., “Introduction to Modern Information Retrieval,” Mcgraw-Hill, New York, 1983.

Schank, R.C., “Conceptualization Underlying Natural Language,” In R.C. Schank and K.M. Colby (Eds), *Computer Models of Thought and Language*, pp. 187-247, W.H. Freeman and Co., San Francisco, 1973.

Schank, R.C., “Conceptual Dependency Theory,” In R.C. Schank, *Conceptual Information Processing*, pp. 22-82, American Elsivier, North-Holland, 1975.

Shankar, AJ. and Yung, Wing, “gNarLi: A Practical Natural Language Interface,” http://www.people.fas.havard.edu/~shankar2/stuff/gnari.html, 2001.

Shimazu, H., S. Arita, and Y. Takashima, "CAPIT: Natural Language Interface Design Tool with Keyword Analyzer and Case-based Parser," *NEC Res. & Develop*, 33:4, 1992, pp.679-688.

Soderland, S.G., *Learning Text Analysis Rules for Domain Specific Natural Language Processing,* Ph.D. Dissertation, 1997, Department of Computer Science, University of Massachusetts Amherst, Amherst, MA.

Templeton, M. and J. Burger, "Considerations for the Development of Natural-Language Interfaces to Database Management Systems," in L. Bolc and M. Jarke (Eds), *Cooperative Interfaces to Information Systems*, pp. 67-99, Springer-Verlag, New York, 1986.

Wald, J.A. and Sorenson, “Resolving the Query Inference Problem using Steiner Trees,” *ACM Transactions on Database Systems*, 9:3, 1984, pp. 348-368.

Waltz, D.L., "An English Language Question Answering System for a Large Relational Database," *Communications of the ACM*, 21:7, 1978, pp.526-539.

Warren, D. and F. Pereira, "An Efficient Easily Adaptable System for Interpreting Natural Language Queries," *Computational Linguistics*, 8:3-4, 1982, pp. 110-122.

Weizenbaum, J., "ELIZA-A Computer Program for the Study of Natural Language Communication between Man and Machine," *Communications of the ACM*, 9:1, 1966, pp. 36-44.

Wilks, Y., “An Intelligent Analyzer and Understander of English,” *Communications of the ACM*, 18:5, 1978, pp. 264-274.

Wood, W.A., “Transition Network Grammars for Natural Language Analysis,” *Communications of the ACM*, 13:10, 1970, pp. 591-606.

Wood, W.A., “Semantics and Quantification in Natural Language Question Answering,” In M. Yovits (ed) *Advanced in Computers*: pp. 1-87, Academic Press, New York, 1978.

Wood, W.A., “Cascaded ATN Grammars,” *American Journal of Computational Linguistics*, 6:1, 1980, pp. 1-15.

Wu, W. and D.M. Dilts, “Integrating Diverse CIM Data Bases: The Role of Natural Language Interface,” *IEEE Trans. Systems, Man, and Cybernetics*, 22:6, 1992, pp.1331-1346.

Zhang, G. Chu, W.W. Meng, F. and Kong G., “Query Formulation from High-level Concepts for Relational Databases,” In N. W. Paton and T. Griffiths (eds), *Prodeedings in User Interfaces to Data Intensive Systems,* pp. 64-74, The IEEE Computer Society, 1999.

Zue, V., “Talking with your Computer,” *Scientific American*, 281:2, 1999, pp. 56-57.