International Journal on Artificial Intelligence Tools
Vol. XX, No. X (2006) 1–8

World Scientific
www.worldscientific.com

A NEW FEASIBLE NATURAL LANGUAGE DATABASE QUERY METHOD

VEERA BOONJING

Mathematics and Computer Science, King Mongkut's Institute of Technology, Ladkrabang, Ladkrabang, Bangkok, 10520, Thailand
kbveera@kmitl.ac.th

CHENG HSU

Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA
hsuc@rpi.edu

Received (February, 2004)
Accepted (March, 2005)

A metadata search approach is proposed to provide practical solutions to the natural language database query problem; where the metadata grow in largely linear manner and the search is linguistics-free. 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.

Keywords: natural language database query, metadata search, Metadatabase, reference dictionary.

1. Asking the database in your own language

Previous natural language database query solutions tend to either lack free-text flexibility (e.g., the concept-form-syntax models) or suffer from implementation complexity (e.g., the linguistics-string models) – see [1]. To illustrate the goals of the new method in this paper, we list below the actual queries that the mocked end users used in a laboratory testing of our prototype.

- Get billing address 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. 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 [2,3]. 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.

2. Making the Approach Work: the Methods and Algorithms

The core logic of the metadata search 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 $\mathbf{I}_{keyword} = (t_1, t_2, \ldots, t_k)$ where t_i is a word/phrase (keyword) in input that also belongs to **K, M or D** of **R**; i = 1, 2, …,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 $\mathbf{I}_{keyword}$ may refer to many elements in **M or D**). Denote this set as V_i. Therefore, we have an ordered k-tuple $\mathbf{V}_{keyword} = (V_1, V_2, \ldots, V_k)$ 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, the metadata search approach works under the necessary condition. Postulate 3 also allows for adding new criteria such as operating rules of the database to enhance the search.

2.1. The Reference Dictionary of Metadata

We use a metadata representation method based on the Metadatabase model [4] to create the reference dictionary. The structure of reference dictionary 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. 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 consists of a problem definition and a solution. A set of keywords and their recognized vertices for a query describes the problem definition and its interpretation defines the solution [1].

2.2. 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. V is a set of vertices of five types: subjects, entities, relationships, attributes, and values; and E is a set of their connection constraints (two owner-member types: subject-(sub)subject-entity/relationship-attribute-value and subject-attribute, and three peer-peer 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.

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 opsI_100|PZ1, V ppsI_54|PZ1, V sfcI_11|PZ1, V sfcI_5|PZ1 }
NAMES	{I opsI_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 The connected feasible graph CFG111 and its query graph QG111 for the feasible graph FG11 are determined, respectively.

2.3. 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 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.

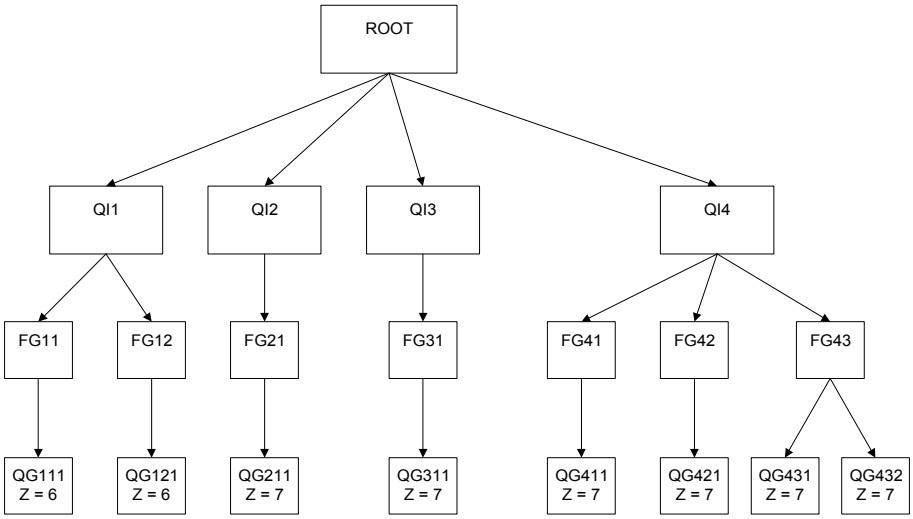

Figure 1: The Query Graph Enumeration Procedure for Query 1.

Note that Figure 1 shows multiple minimum-cost query graphs (interpretations) for Query 1. These minimum-cost query graphs could be equivalent. Query graph a is equivalent to query graph b if any of the following conditions satisfy: (1) entity/relationship solution path of query graph a is equal to that of query graph b; (2) a set of attribute vertices of query graph a is equivalent to that of query graph b; or (3) a set of value vertices of query graph a is equivalent to that 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

The search algorithm itself follows the Branch-and-Bound logic. Given an input $I = (w_1, w_2, \ldots, w_n)$, 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 $I_{keyword}=(t_1, t_2, \ldots, t_k)$ 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 $V_{keyword}=(V_1, V_2, \ldots, V_k)$ 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 $V_{min}=(\text{minV}_1, \text{minV}_2, \ldots, \text{minV}_k)$.

Step 2.2: Determine $V_{minVertexSets}$ of V_{min} such that it contains no minV_i that $|\text{minV}_i|=1$ and its element is an entity/relationship or subject vertex belonging to minV_j where $|\text{minV}_j| = 1$, $i \neq j$, and $j=1,2, \ldots, k$. Therefore, we have $V_{minVertexSets}=(\text{minV}_1, \text{minV}_2, \ldots, \text{minV}_{k^*})$ where $k^* \leq k$. Corresponding to $V_{minVertexSets}$ is $I_{minKeywords}=(t_1, t_2, \ldots, t_{k^*})$.

Step 2.3: Determine a query image set QI such that $QI = \{QI_i \mid QI_i$ is a set of k* recognized vertices, $\{v_{1,i} \in \text{minV}_1, v_{2,i} \in \text{minV}_2, \ldots, v_{k^*,i} \in \text{minV}_{k^*}\}$; $i = 1, \ldots, n$; $n = |\text{minV}_1||\text{minV}_2|\ldots|\text{minV}_{k^*}|$ where $|\text{minV}_j|$, $j = 1, 2, \ldots, k^*\}$.

Step 2.4: Determine a minimal set minQI_i for each $QI_i \in QI$ such that $\text{minQI}_i = \{ v_j \mid$ 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 $j \neq l$; $j, l = 1, \ldots, k^{**}$; $k^{**} \leq k^*\}$.

Step 2.5: Determine a minimal QI set, $QI_{min} = \{ \text{minQI}_i \mid$ for all minQI_i, there does not exist minQI_j such that $\text{minQI}_l=\text{minQI}_j$ where $i \neq j$; $i, j = 1, \ldots, n^*$; $n^* \leq n\}$.

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

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

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.

- 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.

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. 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.

2.4. 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. 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 represents a learning capability to the system. The method 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 and its COSINE measure of similarity. We use a procedure for the COSINE measure to compare a set of keywords representing a query (Q) and a set of keywords representing a case (C) [1].

3. Verifying the Approach in a Laboratory Testing

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. Then, 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 [1] for a complete documentation.)

Analysis I: Truly Natural Query Processing
Result: the software solved every query listed in Section I.
Analysis II: Scalability and Soundness
Result: the software used a limited number of new user-words to solve new queries.
Analysis III: the Necessary Condition and the Sufficient Condition
Result: all queries tested satisfy the necessary condition and the sufficient condition.

References

1. 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.
2. 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
3. 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.
4. 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.