A Branch-and-Bound Algorithm for Natural Language Database Query Using Metadata Search

Veera Boonjing

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

Abstract. This paper presents a branch-and-bound algorithm to process free-text natural language database queries based on the metadata search approach. The approach uses a metadata reference dictionary represented in a semantic graph of enterprise databases. User words, query cases, information models, and database values are vertices of the graph. The paper concludes with possible extensions to offer a full capability of free-text natural language database queries.

1 Introduction

Available results [1] to the problem of free-text natural language database queries have been mostly disappointing. They restrict the syntax of the query or require predefined templates. Therefore, the metadata search approach [2] has been developed as a solution to the problem. The approach employs a metadata reference dictionary represented in a semantic graph of enterprise databases. User words, query cases, information models, and database values are vertices of the graph. The branch-and-bound algorithm interprets a query based on its extracted keywords, corresponding to vertices of the graph. Because keywords may correspond to several vertices of the graph, there could be several different interpretations for the query. The algorithm efficiently searches for the best interpretation for the query.

The remainder of this paper is organized as follows. Section 2 describes the metadata reference dictionary. Section 3 describes the branch-and-bound algorithm. The analysis of the algorithm is given in Section 4. Section 5 concludes with a summary of an incorporation of interactive learning and case-based reasoning into the approach.

2 The Metadata Reference Dictionary

The metadata search approach employs the Metadatabase model [3] as a basis to form the reference dictionary. The benefits of employing the Metadatabase model include its extensibility [3, 4] and its capability to incorporate rules [5, 6, 7] and to support global query processing across multiple databases [8]. The Meatadata model serves as a core structure of the reference dictionary. By using the Two-Stage Entity-Relationship (TSER) modeling method [9], the core structure is expanded to include three additional layers: database values, user-words, and cases and to form an integrated structure of the entire reference dictionary.

A computer-integrated manufacturing (CIM) database is used as an example to demonstrate contents of the reference dictionary. It consists of three systems: an order entry system, a shop floor control system, and a process planning system. Its information models created by using the Two-Stage Entity-Relationship (TSER) method. They become metadata instances stored in certain meta-entities and meta-relationships. The reference dictionary also includes database values, user words, and cases stored along with these information models.

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.

In the reference dictionary of CIM database, the subgraph of G for order processing system is as shown in Figure 1. In this subgraph, S ORDER_PROCESSING - S ORDER-E PART – I opsI_100 part_id – V PZ1 shows owner-member constraint of the type subject - (sub) subject - entity/relationship – attribute - value and E PART - E ORDER_ITEM - E ORDER - E CUSTOMER shows peer-peer constraint of the type entity - entity. The owner-member of the type subject - attribute for this subgraph is shown in Figure 2.

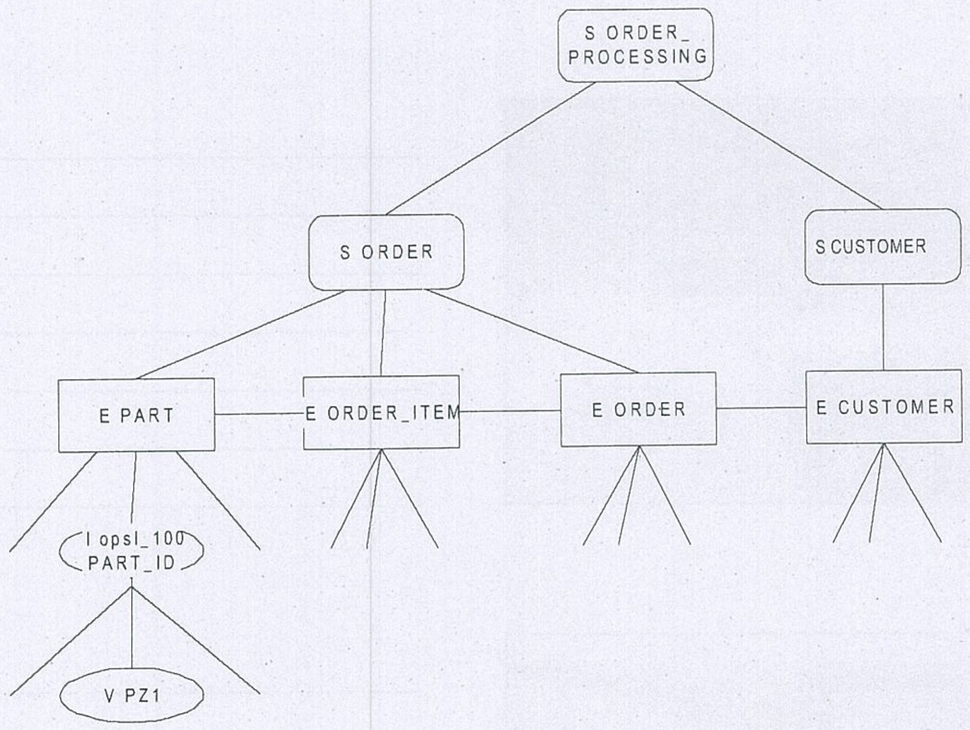

Figure 1: The subgraph of G for order processing system

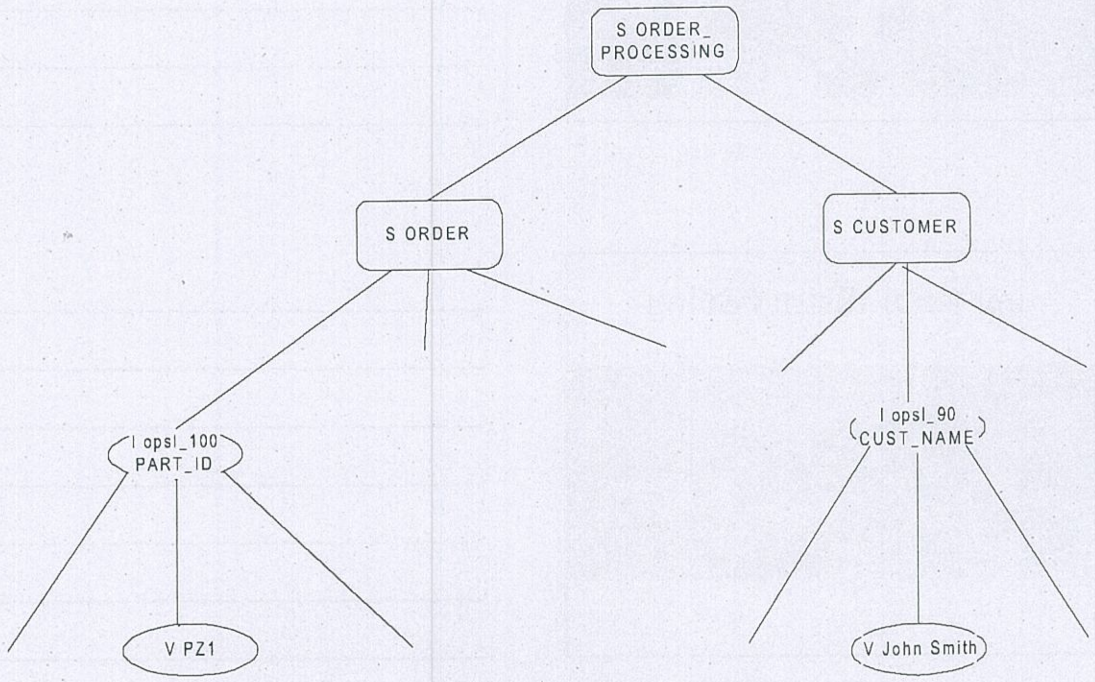

Figure 2: The subgraph of G for order processing system

To demonstrate the interpretation procedure based on the graph G, consider Query 1: "Which customers placed orders on PZ1? Just give me their names." Suppose keywords (words and phrases matching some entries in the reference dictionary or some vertices in graph G) and their recognized vertices (a set of G vertices matching keywords) for this query are as shown in Table 1.

Table 1.Keywords and their recognized vertex sets for query 1

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}

The followings are possible query images generated from Table 1.

QI 1: { E CUSTOMER, S ORDER, V opsl_100|PZ1, I opsl_90 }
QI 2: { E CUSTOMER, S ORDER, V ppsl_54|PZ1, I opsl_90 }
QI 3: { E CUSTOMER, S ORDER, V sfcl_11|PZ1, I opsl_90 }
QI 4: { E CUSTOMER, S ORDER,, V sfcl_5|PZ1 }, I opsl_90 }

Table 2 shows example of semantic paths for recognized vertices of QI1. Note that there are two semantic paths for the recognized vertex V opsl_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.

Table 2. Recognized vertices and their semantic paths.

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

V opsl_100|PZ1	{ (VPZ1, I opsl_100, E PART) }
	{(VPZ1, I opsl_100, E ORDER_ITEM) }

NAMES	{(I opsl_90, E CUSTOMER)}

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

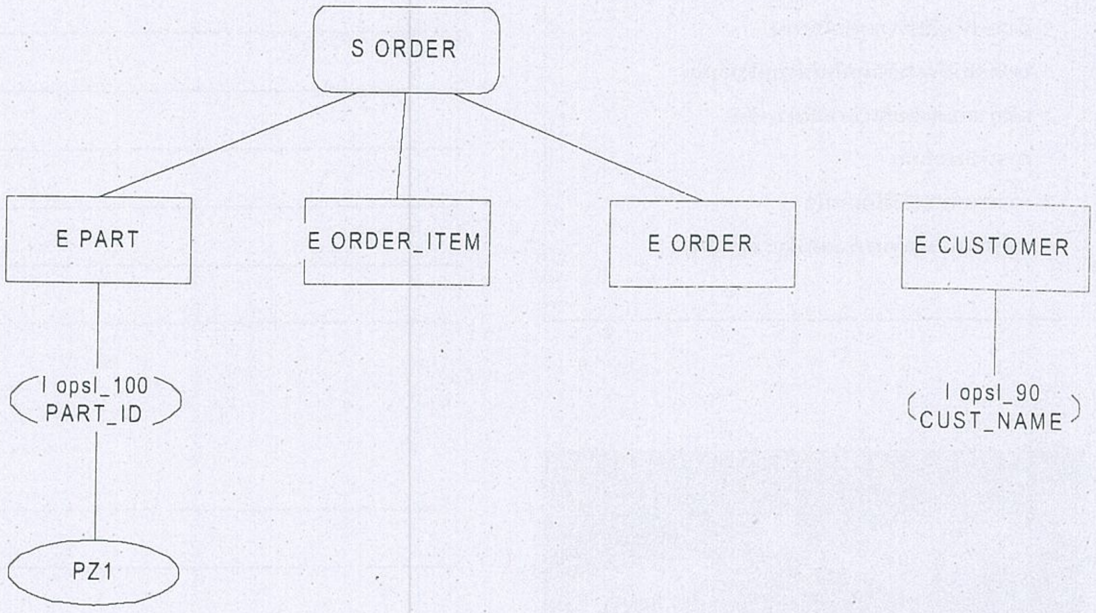

Figure 3: The feasible graph FG11 for QI1.

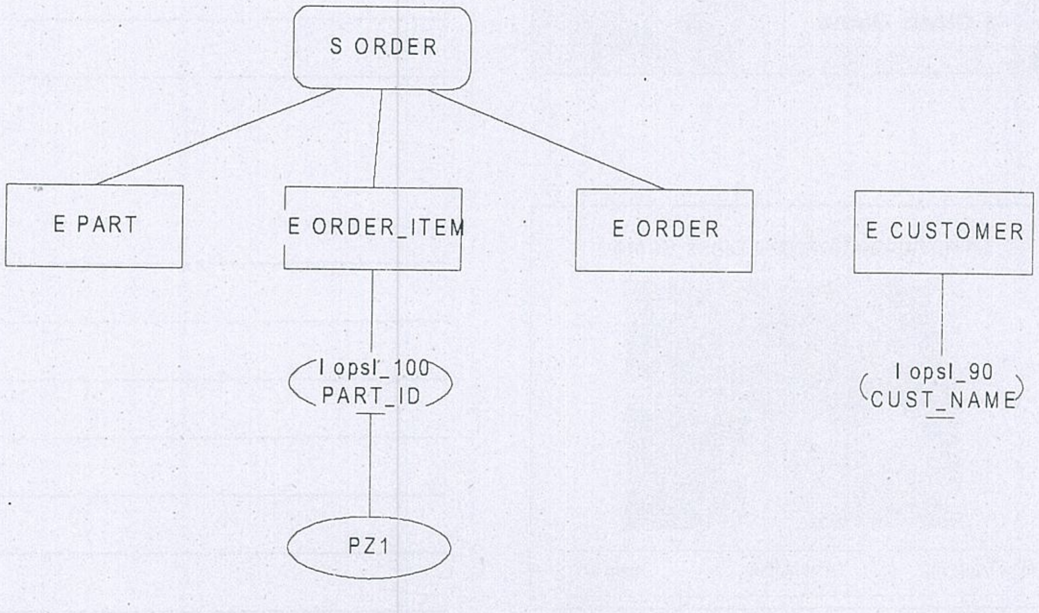

Figure 4: The feasible graph FG12 for QI1.

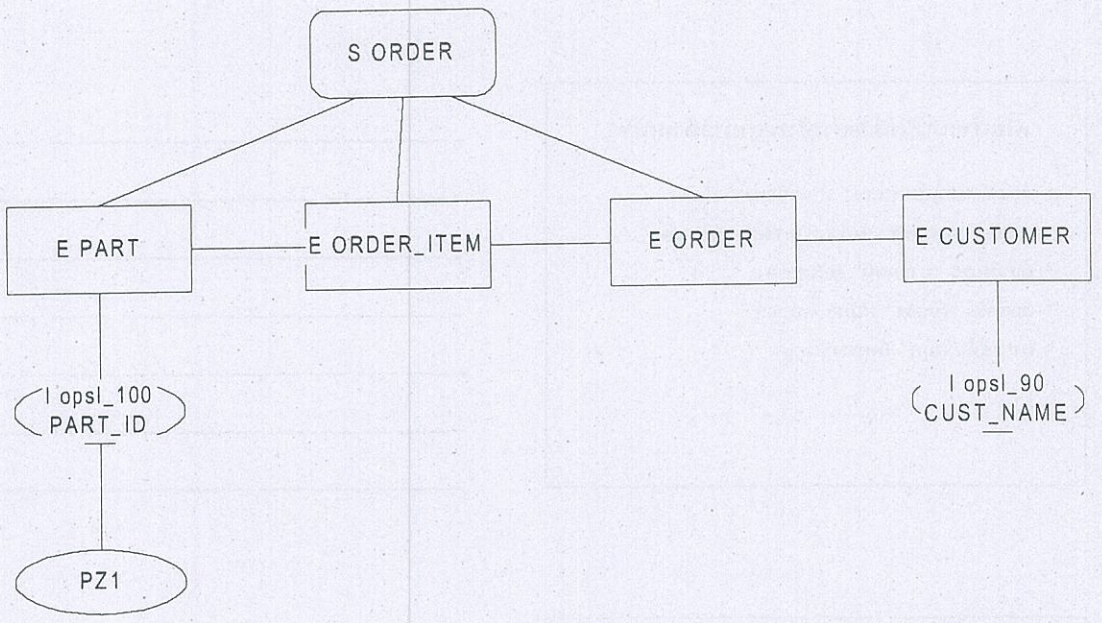

Figure 5: The connected feasible graph CFG111 for FG11.

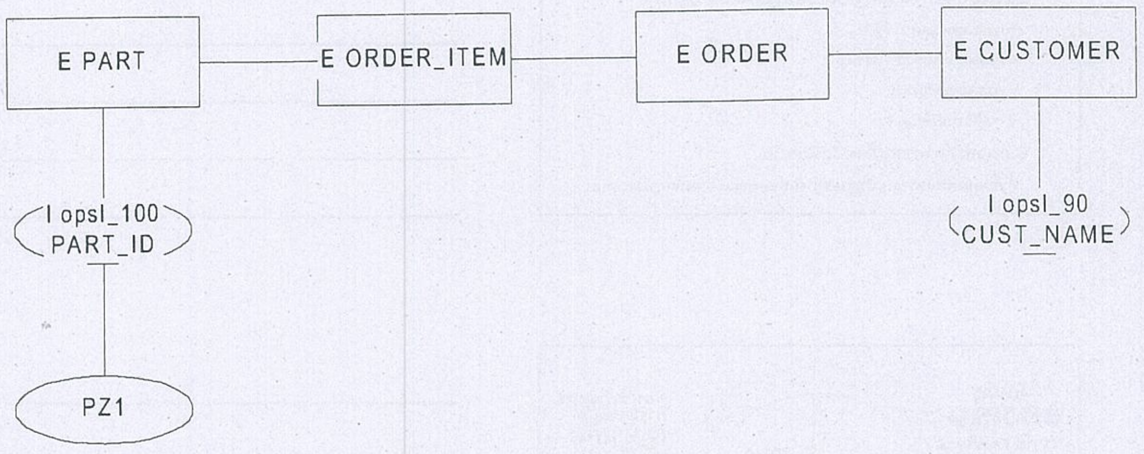

Figure 6: The query graph QG111 of CFG111.

Following the procedures described above, all feasible graphs and query graphs for Query 1 are determined as shown in Figure 7.

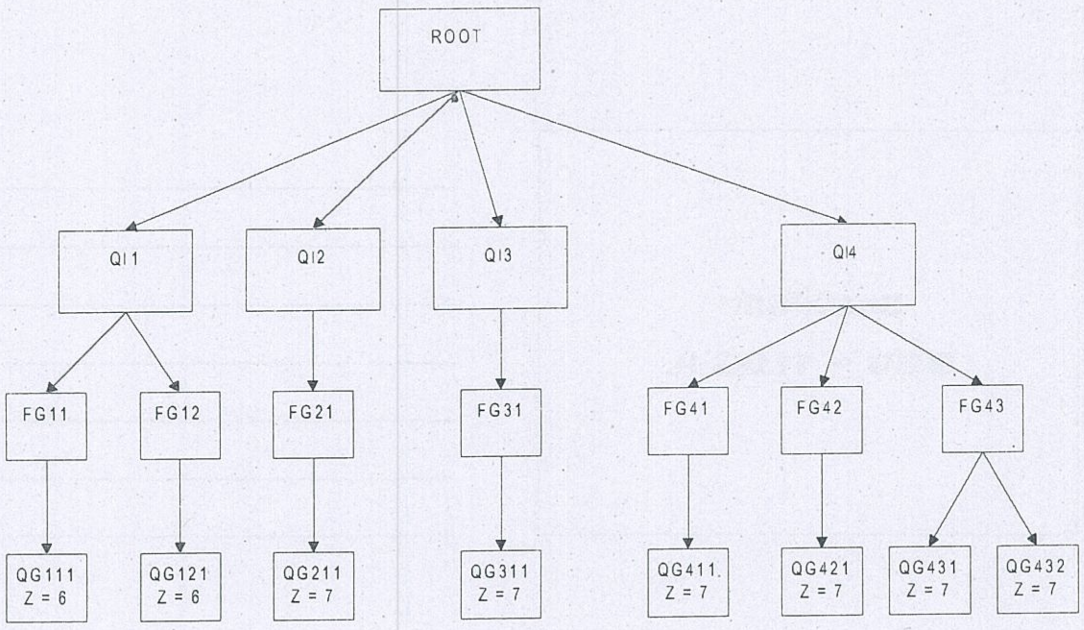

Figure 7: The query graph enumeration procedure for Query 1.

3 The Branch-and-Bound Algorithm

The problem of the branch-and-bound search algorithm is to determine the best query graph (interpretation) among all possible query graphs for a natural language query. The best query graph is the one requiring minimum traversal on the structure of reference dictionary (see [10, 11] for a justification of the objective function). This criterion is used 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. . Costs of enumerated query graphs for Query 1 are shown in Figure 7. The best query graphs for this query is a query graph with cost $z = 6$.

The search problem 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.

When lower bounds of all query images and feasible graphs in Figure 7 are calculated, the complete search graph for Query 1 is shown in Figure 8.

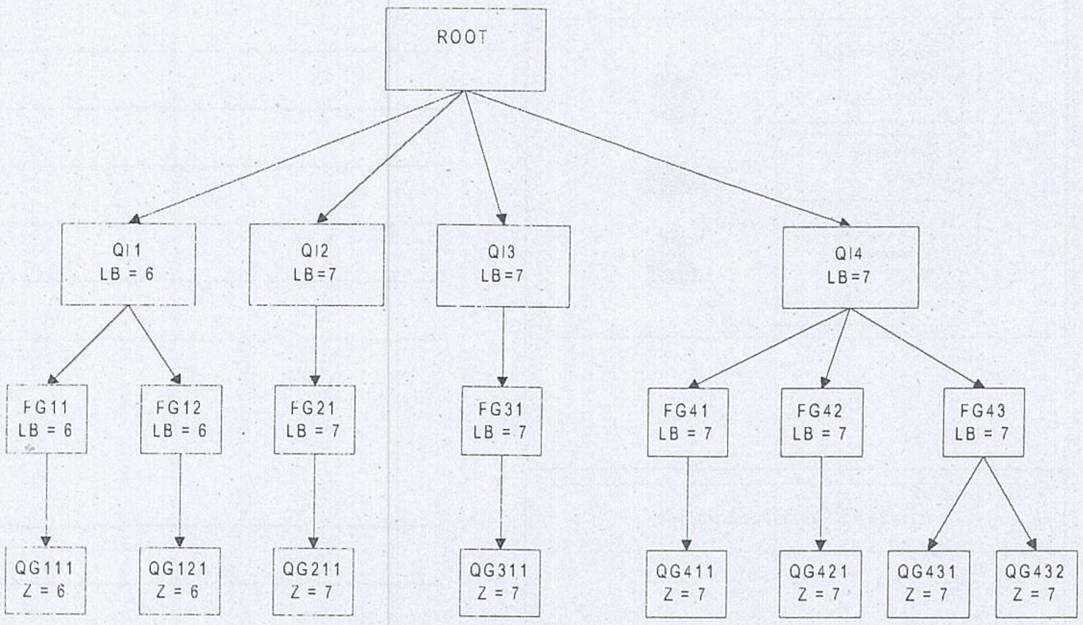

Figure 8: The complete search graph for Query 1.

With a standard branch-and-bound algorithm [12; 13] and the evaluation function LB (), the implicit search graph for Query 1 is as shown in Figure 9.

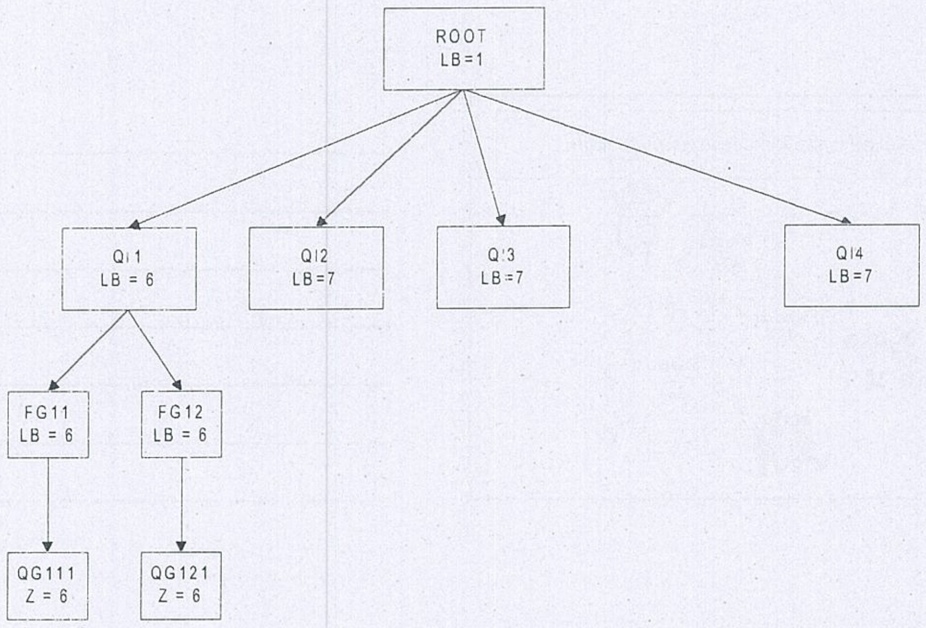

Figure 9: The implicit search graph for Query 1.

Since there exist two minimum-cost query graphs (QG111 in Figure 6 and QG121 in Figure 10) for Query 1. However, they are equivalent according to the following definition.

A query graph 1 is equivalent to a query graph 2 if the following conditions satisfy.

1. Entity/relationship solution path of query graph 1 is equal to entity/relationship solution path of query graph 2.
2. A set of attribute vertices of query graph 1 is equivalent to a set of attribute vertices of query graph 2.
   An attribute set 1 is equivalent to an attribute set 2 if (1) | attribute set 1| = | attribute set 2| and (2) for every element of attribute set 1, there exists one and only one element of attribute set 2 that is equal to (both have the same attribute vertex and the same entity/relationship vertex the attribute belongs to) or is equivalent to (both have the same domain) it.
3. A set of value vertices of query graph 1 is equivalent to a set of value vertices of query graph 2.
   A value set 1 is equivalent to a value set 2 if (1) | value set 1| = | value set 2| and (2) for every element of value set 1, there exists one and only one element of value set 2 that is equal to (both have the same value vertex, attribute vertex, and entity/relationship the attribute belongs to) or is equivalent to (both have the same value and domain) it.

According to this definition, query graph QG111 is equivalent to QG121. Therefore, one of them can be removed. Suppose QG121 is removed, the QG111in Figure 6 is the final interpretation of Query 1. This interpretation is then mapped to the following SQL query. See [2] for mapping rules.

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_I
D and ORDER_HEADER.CUST_ID=CUSTOMER.CUST_ID and
(PART.PART_ID = 'PZ1')

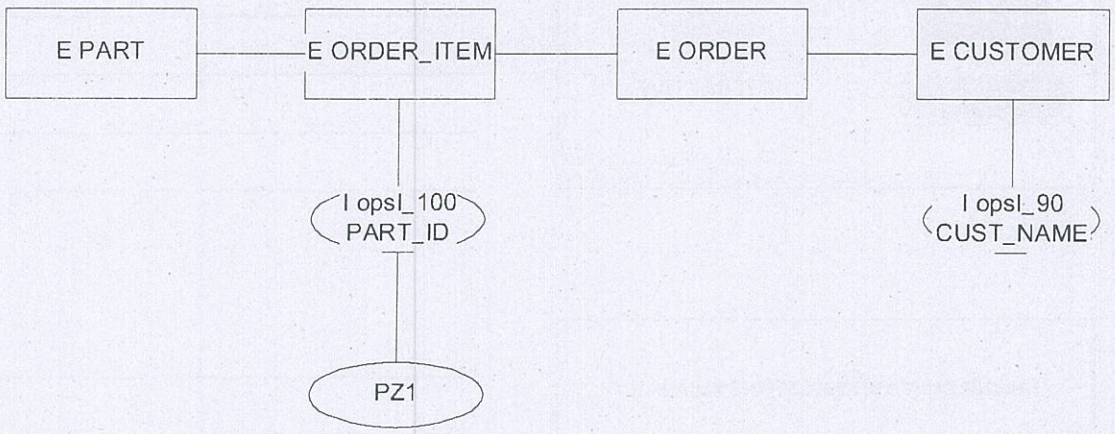

Figure 10: The query graph QG121 of FG12.

4. Analysis

The analysis of the branch-and-bound algorithm is conducted as follows. A correct SQL statement is defined by precise database objects (entities/relationships or tables, attributes or columns, and data base values) and operators. Relaxing the SQL statement by allowing users to use their words/phrases (user-words) to refer to database objects will cause synonyms (where many user-words refers to the same database object) and homonyms (where a user-word refers to many database objects). Synonyms and homonyms introduce the first class of ambiguity called impreciseness. Suppose only synonyms exist, we need a reference dictionary that can be used to map a natural language query containing user-words to the correct SQL statement. The proposed dictionary can be used to accomplish this task. However, there could be homonyms in the reference dictionary. This causes multiple combinations of database objects. The task is to find the best combination. Intuitively, the best combination is the combination that its database objects can be connected without introducing any intermediate database objects. Therefore, the core method uses the "closeness of database objects" as a criterion to choose the best combination. The closeness of a database object combination is measured by counting a number of connections connecting databases objects of that combination.

Until now we have assumed that natural language queries are complete, i.e., there exists a combination of database objects referred to in queries that can be connected without introducing any other database objects. However, we cannot expect users to provide such complete information for all queries. We call this class of ambiguity "incompleteness." Therefore if a query is incomplete in this sense, we need to connect database objects of the query by introducing some intermediate database objects. The graph G of the reference dictionary serves this task. Since homonyms exist in the reference dictionary, a query may have multiple combinations of database objects corresponding to it and database objects in each combination need to be connected. This increases the complexity of the problem, i.e., we need to enumerate all possible combinations, connect their databases objects, count a number of connections for each combination, and look for the best combination – the combination having the fewest connections.

Obviously, the branch-and-bound algorithm can deliver at least one SQL solution to any natural language query containing only one recognized keyword. Therefore, the "necessary condition" to assure an SQL solution exists for a natural language query is that it contains at least one recognized keyword. However, there could be many SQL solutions for such a natural language query. But the more in the number of recognized keywords for a natural language query, the more number of its SQL solutions can be decreased. Therefore, the minimally sufficient set of recognized keywords for the natural language query that can give a single, correct SQL solution for it constitutes the " sufficient condition." In other word, the core method will generate a correct answer as a simple SQL statement if the natural language query contains a complete set of keywords from which, and only from which, a single SQL statement can be constructed to answer the query correctly.

5. Conclusion

The branch-and-bound algorithm is capable of processing free-text natural language inputs under certain conditions (the necessary and sufficient conditions). The necessary condition is the text input contains at least one recognized keyword (a keyword found in the reference dictionary). The sufficient condition is the text input contains a complete set of keywords from which, and only from which, a single SQL query can be constructed to answer the query correctly.

The branch-and-bound algorithm provides a good basis for immediate extensions to develop the full capability of free-text natural language query. These extensions include two significant research opportunities: interactive learning and case-based reasoning.

The purpose of interactive learning is to walks users to the kind of feedback it needs to identify the correct solution for a query. Its tasks are to interact with users to confirm the results with them, to ask them for its information needed, and to ask them to pick their intended solution. These tasks need good dialogues controlling overall interacting process. Therefore, the future research suggestion is in designing such controlled dialogues for this interactive learning. The interactive learning also includes interaction with users to acquire new user-words.

The case-based reasoning interprets natural language queries by finding and using the most similar past case. If a perfectly matched case for a query does not exist but there exists a case for the query that is significantly similar (say at least 60%), the case-based reasoning must modify (reuse) the case solution (query graph) to obtain a solution. Therefore, the basic challenge here is how to modify the case solution to obtain the solution for the query.

References

1. Androutsopoulos, I., Ritchie, G.D., and Thanisch, P. 1995. Natural language interfaces to databases – An introduction, Journal of Natural Language Engineering Vol. 1 No. 1: pp. 29-81.
2. Boonjing, V. and Hsu C. 2002. Metadata Search: A New Approach to Natural Language Database Interfaces. Proceedings IASTED International Conference on Information Systems and Databases. Tokyo, Japan.
3. Hsu, C. Bouziane, M. Ratter, L. and Yee, L. 1991. Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach. IEEE Trans. on Software Engineering Vol. 17 No. 6: pp. 604-625.
4. Hsu, C. 1996. Enterprise Integration and Modeling: the Metadatabase Approach. Boston: Kluwer Academic Publishers.
5. Bouziane, M. and Hsu, C. 1993. A Rulebase Model for Data and Knowledge Integration in Multiple Systems Environments. J. Artificial Intelligence Tools Vol.2 No.4: pp. 485-509.
6. Bouziane, M. and Hsu C. 1997. A Rulebase Management System Using Conceptual Modeling. J. Artificial Intelligence Tools Vol.6 No.1: pp. 37-61.
7. Babin, G. and Hsu C. 1996. Decomposition of Knowledge for Concurrent Processing. IEEE Trans. Knowledge and Data Engineering Vol.8 No.5: pp. 758-772.
8. Cheung, W. and Hsu, C. 1996. The Model-Assisted Global Query System for Multiple Databases in Distributed Enterprises. ACM Trans. on Information Systems Vol. 14 No. 4: pp. 421-470.
9. Hsu, C. Tao Y., Bouziane M., and Babin G. 1993. Paradigm Translations in Integrating Manufacturing Information Using a Meta-Model: the TSER Approach. J. Information Systems Engineering Vol.1 No.1: pp325-352.
10. Johnson, J.A. 1995. Semantic Relatedness. Computers Math. Applic, Vol. 29 No.5: pp. 51-63.
11. Wald, J.A. and Sorenson. 1984. Resolving the Query Inference Problem using Steiner Trees. ACM Transactions on Database Systems Vol. 9 No. 3: pp. 348-368.
12. Kumar, V. 1987. Branch-and-Bound Search. In S.C. Shapiro (ed), Encyclopedia of Artificial Intelligence. New York: Wiley-Interscience.
13. Nau, D.S. Kumar, V. and Kanal, L. 1984. General Branch and Bound, and its Relation to A* and AO*. Artificial Intelligence Vol.23: pp.29-58.