International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:11, 2009

Natural Language Database Interface for Selection of Data Using Grammar and Parsing

N. D. Karande, and G. A. Patil

***Abstract*—**Databases have become ubiquitous. Almost all IT applications are storing into and retrieving information from databases. Retrieving information from the database requires knowledge of technical languages such as Structured Query Language (SQL). However majority of the users who interact with the databases do not have a technical background and are intimidated by the idea of using languages such as SQL. This has led to the development of a few Natural Language Database Interfaces (NLDBIs). A NLDBI allows the user to query the database in a natural language. This paper highlights on architecture of new NLDBI system, its implementation and discusses on results obtained. In most of the typical NLDBI systems the natural language statement is converted into an internal representation based on the syntactic and semantic knowledge of the natural language. This representation is then converted into queries using a representation converter. A natural language query is translated to an equivalent SQL query after processing through various stages. The work has been experimented on primitive database queries with certain constraints**.**

***Keywords*—**Natural language database interface, representation converter, syntactic and semantic knowledge.

I. Introduction

IN Natural Language Database Interface (NLDBI), manual construction of translation knowledge normally undermines domain portability because of its expensive human intervention. To overcome it, the work carried out linguistically motivated database semantics in order to systematically construct translation knowledge [1]. Database semantics consists of two structures; first one is designed to function as a translation dictionary and other one to contain selection restriction constraints on domain classes. The database semantics is semi-automatically obtained from a semantic data model for a target database. Based on this database semantics, a conceptual NLDBI translation scheme is developed. Translating a natural language question into a database query suffers from translation ambiguity problem. In NLDBI, translation ambiguities occur when a linguistic term is associated with two or more domain classes. That is, a linguistic term has many translation equivalents in physical database structures. In previous works, translation ambiguity is not seriously considered, because it is assumed that, given a specific database domain, each domain terminology has a unique domain class. In experimental systems, this assumption can be true. However, in real practical databases, this assumption is too strong.

One of the earliest NLDBI systems was LUNAR [2] which was built on top of a database of rock samples brought back from the Apollo missions to the moon. It uses an augmented transition network (ATN) parser, a popular parser for computational linguists. A query is matched recursively in a semantic interpretation module to produce a representation that together with quantifier information is ordered using various heuristics. The final result is a representation language in a logical form [3]. LUNAR and other early NLIDBs were application dependent. Because of this, although the prototype worked well, it was not very portable in the sense that major modifications were required to use the NLIDB for different databases. English Wizard is another successful natural language query tool for relational database. It is one of the leading software products that translate ordinary English database requests into Structured Query Language (SQL), and then return the results to the client. English Wizard enables most database reporting tools and client/server applications to understand everyday English requests for information, and also provides graphical UI.

All these tools translate a natural language query into an intermediate language similar to first order logic and then into SQL using a set of definitive rules. The intermediate language expresses the meaning of the query in terms of high-level concepts that are independent of database structures. In the translating process, the premises of rules must match with phrases of the query exactly; otherwise the rule will be re-jected. Our goal is to overcome this problem by constructing the most probable grammar tree and analyzing the non-terminals (phrases) in the grammar tree to collect the parameters, which will be used in SQL.

II. Natural Language Database Interface Based On Grammar

The system architecture of natural language database interface developed is given in Fig. 1, which depicts the layout of the processes included in converting NL query into a syntactical SQL query to be fired on the RDBMS.

N. D. Karande is with the Department of Computer Science & Engineering, Shivaji University, Kolhapur 416113, (MS), India (Corresponding author to provide phone: 91-9823986827; e-mail: nikhilkarande18@gmail.com).
G. A. Patil is with the Department of Computer Science & Engineering, Dr. D. Y. Patil Colloge of Engineering, Kolhapur 416006, (MS), India (e-mail: gasunikita@yahoo.com).

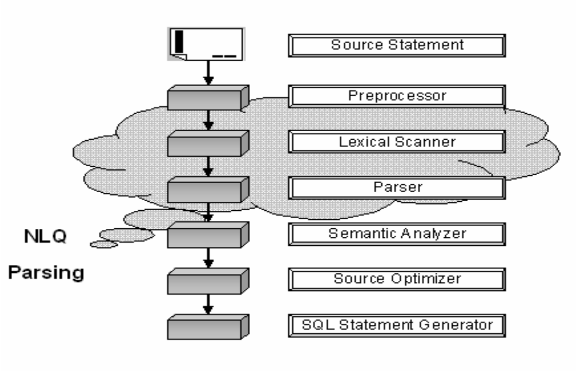


Fig. 1 Architecture of NLDBI System.

To process a query, the first step is speech tagging; followed by word tagging. The second step is parsing the tagged sentence by a grammar. The grammar parser analyzes the query sentence according to the tag of each word and generates the grammar tree/s. Finally, the SQL translator processes the grammar tree to obtain the SQL query.

The paper is based on a unique concept of processing user natural language into a technical form so as to access the data from higher end data storage. NLDBI is a system that allows users to access a database in natural language and has been a popular field of study. Suppose we consider a properly normalized database. Now if the user wishes to access the data from the table, he/she accesses the tables in his/her language.

III. Grammar and Parsing

Consider a sentence w_{1m} which is a sequence of words w_1 w_2 $w_3 \ldots w_m$ (ignoring punctuations), and each string w_i in the sequence stands for a word in the sentence. The grammar tree of w_{1m} can be generated by a set of predefined grammar rules; usually more than one grammar tree may be generated. The formalizing capability of grammar help in describing most sentence structures and built efficient sentence parsers.

A parser is one of the components in an interpreter or compiler, which checks for correct syntax and builds a data structure (often some kind of parse tree, abstract syntax tree or other hierarchical structure) implicit in the input tokens. The parser often uses a separate lexical analysis to create tokens from the sequence of input characters. Parsers may be programmed by hand or may be semi automatically generated (in some programming language) by a tool (such as Yacc) from a grammar written in Backus-Naur form.

The SQL translator generates query in SQL. Using grammar the parse tree is obtained from the input statement. The leaves of the parse tree are translated to corresponding SQL. Fig. 2 depicts the processing of English input statement to generate SQL query. The entire process involves tagging of input statement, apply grammar and semantic representation to generate parse tree, analyze the parse tree using grammar and translating the leaves of the tree to generate corresponding SQL query.

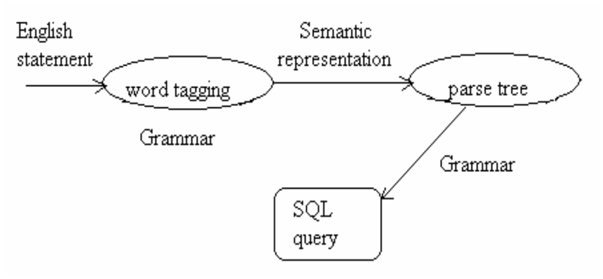


Fig. 2 Generation of SQL query from English Statement.

The database tables considered are EMP (empid, empname, salary, edepid, address, post, mobileno), DEPT (deptid, deptname, deptloc, dcapacity) and PROJECT (pid, pname, epid). From the input NL statement, to generate parse tree the grammar written based on database tables is:

WhatKeyBank → for | of | with | is | where | whose | having | in | on
AAnTheBank → a | an | the
empid → integer | id | number
empname → string | name
salary → integer | salary | income | earning
mgrid → integer | manager | boss | superior
edeptid → integer | id | number
deptid → integer | id | number
deptname → string | name
deptloc → string | location
dcapacity → integer | capacity
EmpTable → employee | worker | person | emp | employees | emps | workers |persons
ProjectTable → project | projects
DeptTable → department | dept | dpt | departments | depts | dpts

The experimental work is to design an interface for generating queries from natural language statements/ questions. It also consists of designing a parser for the natural language statements, which will parse the input statement, generate the query and fire it on the database. The experimental work will understand the exact meaning the end user wants to go for, generate a *what- type* sentence and then convert it into a query and handover it to the interface. The interface further processes the query and searches for the database. The database gives the result to the system which is displayed to the user. The following modules were developed.

- An Interface: It allows the user to enter the query in NL, interact with the system during ambiguities and display the query results.
- Parsing: Derives the Semantics of the statement given by the user and parses it into its internal representation, to convert NL input statement into what- type question for selection of data.

- Query Generation: It generates a query against the user statement in SQL and passes on to the database.

The algorithm designed is put as mention below:

Algorithm 1 Generation of parse tree/s from NL statement using grammar.

```
1.  Read Input Statement S
2.  for each word $W_i$ from S do
3.        if ($W_i \in$ Grammar G) then
4.              Add $W_i$ to Symbol Table ST
5.        end if
6.  end for
7.  for each $W_i$ from ST do
8.        Add $W_i$ to parse tree/s T for What- type question/s
9.  end for
10. Display What- type question/s Q
11. Read Input Q
12. for each $W_i$ from Q do
13.       if ($W_i \in$ G) then
14.             Add $W_i$ to parse tree for SQL- query
15.       end if
16. end for
17. Display SQL- query
```

A. *Scope of the Experimental Work*

1 To work on a Relational database (RDBMS), one should know the syntax of the commands of that particular database software.

2 The Natural language processing is done on statements written in English language.

3 NL Input from the user is converted in the form of *what- type* questions only.
For example: What is salary of employee with name Nikhil

4 A limited Data Dictionary is used where all possible words related to a particular system are included. The Data Dictionary of the system need to be regularly updated with words that are specific to the particular system.

5 Ambiguity among the words is taken care of while processing the natural language.

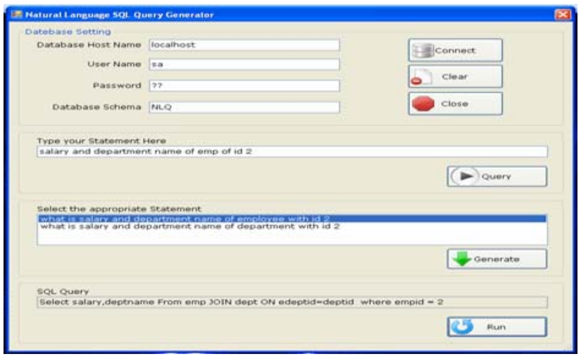
6 All the names in the input natural language statement have to be in double quotes.
For Example: Address of emp “Vivek”

7 Data dictionary used are: EMP, DEPT and ROJECT

IV. Experimental Results

The system implemented was tested for variety of NL statements under various categories and the results obtained were satisfactory under the known constraints. The results were categorized based on the generation of unambiguous pares tree, ambiguous parse trees with two and three parse trees.

A. *Generation of ambiguous parse trees.*



Database Host Name: localhost
User Name: sa
Password: ??
Database Schema: NLQ
Type your Statement Here: salary and department name of emp of id 2
Select the appropriate Statement:
what is salary and department name of employee with id 2
what is salary and department name of department with id 2
SQL Query: Select salary,deptname From emp JOIN dept ON edeptid=deptid where empid = 2

Fig. 3 NLDBI System.

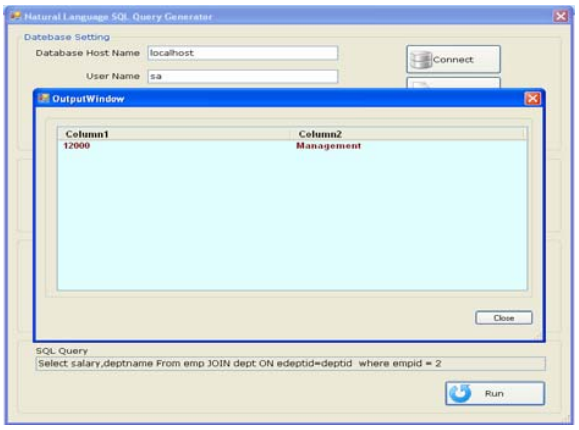
The Fig. 3 shows the typical category of generating ambiguous parse trees.

1 The user expects the salary and department name of employee with id 2 and accordingly the statement that he gives to the system may be as under;
“Salary and department name of emp of id 2”

2 The result generated depicts the ambiguous parse trees were the system is not able to identify the expected meaning of the statements. Instead it generates more than one parse trees leading two different meanings.
For example:
i What is salary and department name of employee with id 2
ii What is salary and department name of department with id 2

3 The user here can interact to remove the ambiguity by choosing the appropriate options.

4 The SQL query is generated by the system which further fired on to the database to obtain the results as shown in Fig. 4 as employee salary – “12000” and department name – “Management”.



Database Host Name: localhost
User Name: sa

Column1	Column2
12000	Management

SQL Query: Select salary,deptname From emp JOIN dept ON edeptid=deptid where empid = 2

Fig. 4 Result for NL statement input to system.

V. Conclusion

The NL statement is converted into machine understandable form such as SQL. The NLDBI system is tested for more than 75 different NL input statements and the system works satisfactorily. The advantage of NLDBI system is that it works on a Relational database and removes ambiguities. So far, our NLDBI system considers *selection of data* and performing primitive queries onto the database and JOIN operation with some constraints. The next step of research is to optimize grammar to accommodate more complex queries.

Acknowledgment

I express my deep sense of gratitude and appreciation towards my research guide Prof. G. A. Patil for his continuous inspiration and valuable guidance in throughout my dissertation work.

References

[1] In-Su Kang, Jae-Hak J. Bae, Jong-Hyeok Lee "*Database Semantics Representation for Natural Language Access.*" Department of Com Computer Science and Engineering, Electrical and Computer Engineering Division Pohang University of Science and Technology (POSTECH) and Advanced Information Technology Research Center (AITrc), 2002.

[2] Woods, W., Kaplan, R. *"Lunar rocks in natural English: Explorations in natural language question answering."* Linguistic Structures Processing. In Fundamental Studies in Computer Science, 5:521-569, 1977.

[3] Androutsopoulos, I., Richie, G.D., Thanisch, P. *"Natural Language Interface to Database – An Introduction."* Journal of Natural Language Engineering, Cambridge University Press. 1(1), 29-81, 1995.

[4] Linguistic Technology. English Wizard – Dictionary Administrator's Guide. Linguistic Technology Corp., Littleton, MA, USA, 1997.

[5] Dan Klein, Christopher D. Manning: Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. ACL 2004: 478-485.

[6] M-C.de Marneffe, B. MacCartney, and C. D. Manning. "Generating Typed Dependency Parses from Phrase Structure Parses." In Proceedings of the IEEE /ACL 2006 Workshop on Spoken Language Technology. The Stanford Natural Language Processing Group. 2006.

[7] Dan Klein and Christopher D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10.

[8] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In LREC 2006.

N. D. Karande received the B.E. degree in Computer Science and Engineering from Bharati Vidyapeeth College of Engineering, Kolhapur, India in 2006. He is doing his MTech in Computer Science and Technology at Shivaji University, Kolhapur, India. From 2007 to 2009, he is working as Lecturer at Bharati Vidyapeeth College of Engineering, Kolhapur, India. He has published various papers in the area of Network Security and Natural Language Processing.

G. A. Patil received the BE degree in Computer Science and Engineering from PES college of Engineering, Mandya and ME in Computer Science and Engineering from Walchand College of Engineering, Sangli. Since 1992 he is working as Assistant Professor at D. Y. Patil College of Engineering, Kolhapur, India. He has published various papers in the area of Distributed System and Information Security.