ISSN No. 0976-5697

Volume 5, No. 6, July-August 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Natural Language Interface to Databases – An Implementation

Er. Amit Chaudhary[1], Er. Annu Battan [2]
Assistant Professor, CSE dept.[1], Student, M.Tech-CSE[2]
[1,2]Modern Institute Of Engineering & Technology, Mohri Kurukshetra, India

Abstract: Information can be stored in a computer file to make a list easier and faster to use. Such a file is called a database. Database Management Systems (DBMS) have been widely used for storing and retrieving data. However, databases are commonly hard to use because storing and retrieving the information from database requires the knowledge of database language like SQL. Structured Query Language (SQL) is standard for accessing and manipulating the information stored in database. But everybody not has known about the structure and syntax of SQL. So NLIDB is developed that use natural language and convert this NL to SQL Query. The idea of using NL (natural language) has prompted the development of new type of processing method called Natural Language Interface to Database systems (NLIDB). So everybody retrieve information from database easily without any knowledge of SQL queries.

Keywords: Natural Language Interface to Database, Hindi Language Interface to Database, Lexicon, Tokenizer, Structured Query Language, Database Management Systems

I. INTRODUCTION

The proposed system converts the Hindi language into SQL query. To achieve the goal of NLIDB a step by step methodology has been given below:

a) Create STUDENT database which store information about students, all the entries are in Hindi language.
b) Create a database which contain all the table name, columns name, conditions, operators and all the tokens which can be given in query and their corresponding English word.
c) Tokenize the input Hindi sentence given as a query by user.
d) Identify the nature of query.
e) Extract the table name, column names, conditions, functions, operators included in input query by mapping tokens with the database values.
f) Generate the SQL query from the extracted values.
g) Execute SQL query and give output to user in Hindi language.

II. LITERATURE REVIEW

Prototype for NLIDB had appeared in late sixties and early seventies. Since then a number of systems have developed.

The best known NLIDB of sixties and early seventies was LUNAR , It answers the questions about samples of rocks brought back from the moon [6]. The meaning of the system name is in relation to the moon. LUNAR system has two databases, one is chemical analysis and other is literature references. The LADDER (Language Access to Distributed Data with Error Recovery) [5] was designed for US Navy ships with a natural language interface. It takes queries in English language. The system was designed as a management aid to navy decision makers. CHAT-80 [2] system came in eighties and was implemented in prolog. TEAM was developed in 1987.TEAM was designed to be easily configurable by database administrators with no knowledge of NLIDBs [3] [4]. GINLIDB The system was developed in 2009 and is called a Generic Interactive Natural Language Interface to Database. It is designed by the use of UML and developed using visual basic .NET 2005.The system includes two main components. 1) Linguistic handling component and 2)SQL constructive component [1] . WASP (Word Alignment-based Semantic Parsing) is a system developed at the University of Texas, Austin by Yuk Wah Wong [7]. The system was designed for achieving the goal of constructing" a complete, formal, symbolic, and meaningful representation of a natural language sentence", that can also be executed to the NLIDB domain. Prolog was used as the formal query language [7].

III. SYSTEM ARCHITECTURE

The architecture of interface to database using Hindi language is composed of four phases. The phases are given below.

a) To tokenize the input Hindi sentence.
b) Map the tokens with lexicon which store all the tokens and their corresponding English word.
c) Formulate SQL query with the help of query generator.
d) Execute the query and display result on interface to user.

Output of one phase is given to next phase. To map the tokens database is used which store all usable tokens. Architecture of system is given in figure 1.

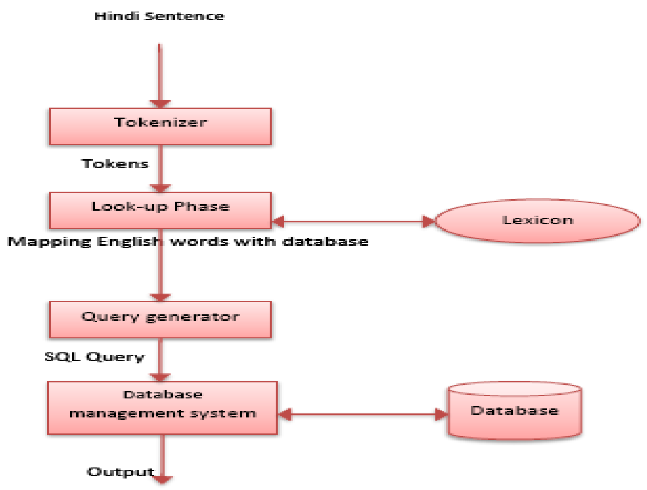


Figure 1: Architecture of the system

IV. HINDI LANGUAGE INTERFACE TO STUDENT DATABASE MANAGEMENT SYSTEM

For case study of interface to database using Hindi language use STUDENT database. This database has two tables STUDENT and DEPARTMENT.STUDENT table contain information about students. It has attributes roll no (roll number), name, city, dob (date of birth), dept_no, Marks. roll no is primary key, no two students can have same roll no.

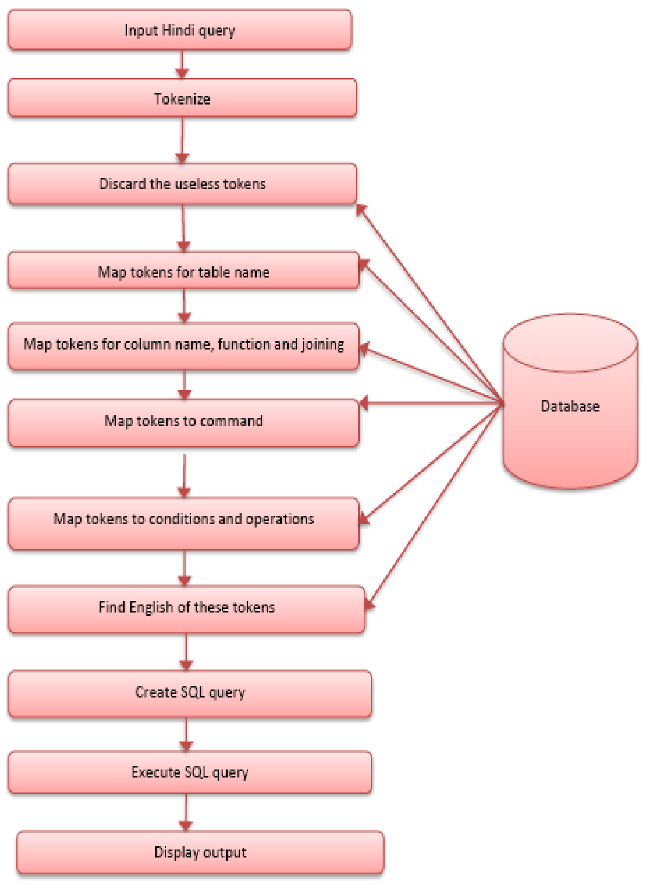

Input Hindi query
Tokenize
Discard the useless tokens
Map tokens for table name
Map tokens for column name, function and joining
Map tokens to command
Map tokens to conditions and operations
Find English of these tokens
Create SQL query
Execute SQL query
Display output
Database

Figure 2: Workflow of the system

A. *Input Hindi query:*

Query given by user is a Hindi sentence. That is composed of Hindi words that we call tokens. Here we explain the working of system with the help of an example sentence given below.
Example sentence

उन सभी विधार्थीयो का नाम ,अंक बताओ जिनका शहर ' कुरुक्षेत्र ' के बराबर और अंक 75 से ज्यादा या बराबर हो

B. *Tokenizing:*

Tokens are gathered from the input sentence by using the logic that all the tokens are separated by a gap from their adjacent tokens. The above sentence has 20 tokens. Some of them are useless that have no further use.

उन, सभी, विधार्थीयो, का, नाम, अंक, बताओ, जिनका, शहर, कुरुक्षेत्र , के, बराबर, और, अंक, 75, से, ज्यादा, या, बराबर, हो

C. *Discard the useless tokens :*

To find out the nature of tokens whether it is table name, field name, condition, command, operation or any useless token we create a database TOKEN which has all the tokens that can be given by user for our STUDENT database. This database has four fields id, Hindi_token, English_token, type. Type tells about the nature of token.

The TOKEN table consists of four columns. Type column tell about the type of token.

We divide the tokens into 11 categories. 1) selection_start(उन, सभी, वो, उस) 2) field_start (के, का, की) 3) Command 4) Table_name **(**two tables STUDENT and DEPARTMENT) 5) Field (name. roll no, city, name, dob, marks, dept_no and dept_name) 6) Logop (AND, OR and NOT) 7) Cond_start (जिनकी,जिनका, जिसका) 8) Function (min(), max(), sum(), avg()) 9) Condop (ज्यादा या बराबर. कम, ज्यादा, बराबर, कम या बराबर) 10) Cond_op_start (के, से) 11) Cond_end (है, हो)

D. *Map tokens to table name:*

In example query 'विधार्थीयो' is the table_name.

Table_name.english = STUDENT. So the STUDENT is saved as table name.

E. *Map tokens for Function, Fields and Joining:*

In our example query we have two columns one is 'नाम' and other is 'अंक'. Output of this phase will be
table_name.column_name.english
STUDENT.name, STUDENT. marks

For input sentence we have no need of joining because both the columns name and marks belong to same table STUDENT.

F. *Map the token with command name:*

Command name in input example query is 'बताओ'. We store the equivalent English word for this in command string. Therefore command.english = select.

G. *Map token for condition and operations :*

For our example query 'शहर ' कुरुक्षेत्र ' के बराबर और अंक 75 से ज्यादा या बराबर' are stored in temp_list. To solve the condition we have need of four arrays list to store columns, value, conditions and logical operators. Match the first token of temp_list with field type tokens. As we know that condition would have definitely columns so if matching is true we store column in column_arraylist after conversion into English. It also checks whether column belongs to same table or not as we did above and removes the token from temp_list.

In this example sentence these array list have following values.

Column_arraylist- city, marks
Condition_arraylist- =, >=
Logicalop_arraylist- and
Value_arraaylist- कुरुक्षेत्र, 75

Therefore output of this phase is

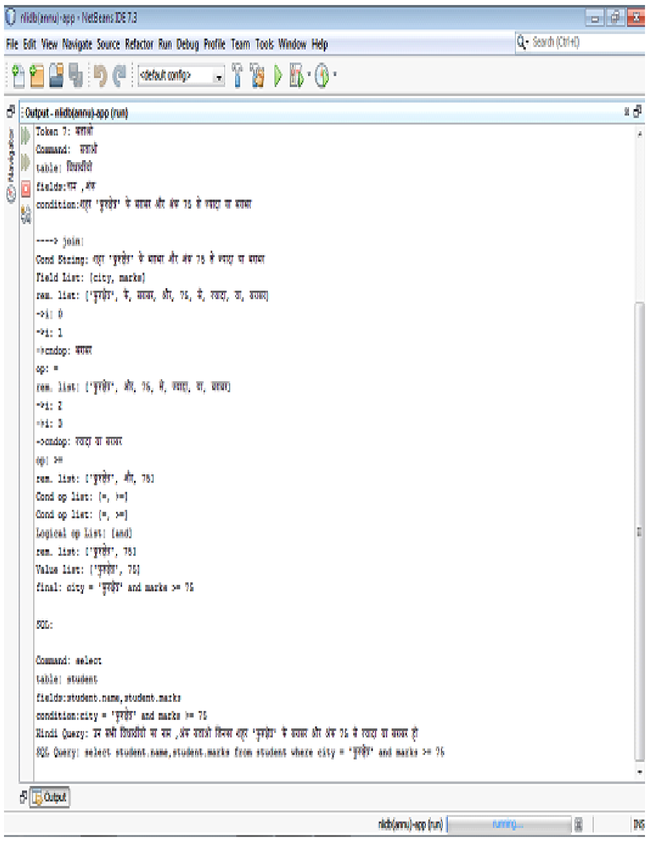
city = 'कुरुक्षेत्र' and marks >= 75

H. Create SQL query:

In this phase SQL query is created with the help of output of various phases given above. We have table name, columns name, command, condition and operations. Now we create he SQL query in this way

"command" "column(s) name" from "table_name" where "condition"

Therefore for our example sentence SQL query is Select student.name,student.marks from student where city = 'कुरुक्षेत्र' and marks >=75

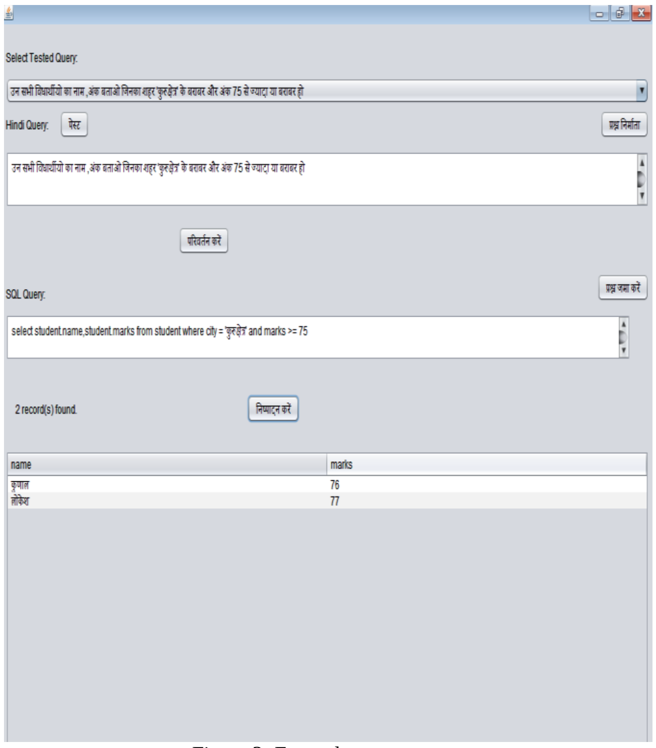

nlidb(annu)-app - NetBeans IDE 7.3
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Output - nlidb(annu)-app (run)
Output
nlidb(annu)-app (run)
running...

```
Token 7: बताओ
Command:  बताओ
table: विद्यार्थी
fields:नाम ,अंक
condition:शहर 'कुरुक्षेत्र' के बराबर और अंक 75 से ज्यादा या बराबर

----> join:
Cond String: शहर 'कुरुक्षेत्र' के बराबर और अंक 75 से ज्यादा या बराबर
Field List: [city, marks]
rem. list: ['कुरुक्षेत्र', के, बराबर, और, 75, से, ज्यादा, या, बराबर]
->i: 0
->i: 1
->cndop: बराबर
op: =
rem. list: ['कुरुक्षेत्र', और, 75, से, ज्यादा, या, बराबर]
->i: 2
->i: 3
->cndop: ज्यादा या बराबर
op: >=
rem. list: ['कुरुक्षेत्र', और, 75]
Cond op list: [=, >=]
Cond op list: [=, >=]
Logical op List: [and]
rem. list: ['कुरुक्षेत्र', 75]
Value list: ['कुरुक्षेत्र', 75]
final: city = 'कुरुक्षेत्र' and marks >= 75

SQL:

Command: select
table: student
fields:student.name,student.marks
condition:city = 'कुरुक्षेत्र' and marks >= 75
Hindi Query: उन सभी विद्यार्थीयो का नाम ,अंक बताओ जिनका शहर 'कुरुक्षेत्र' के बराबर और अंक 75 से ज्यादा या बराबर हो
SQL Query: select student.name,student.marks from student where city = 'कुरुक्षेत्र' and marks >= 75
```

Figure 2: Step by step conversion into SQL

I. Execute SQL query:

The SQL query is executed on the database. The output is provided to user on user interface in Hindi language. For the example query output is the name and marks of all the student whose city is 'कुरुक्षेत्र'and they scored 75 or more than 75 marks. The exact output is shown below in the figure 3.



Select Tested Query: उन सभी विद्यार्थीयो का नाम ,अंक बताओ जिनका शहर 'कुरुक्षेत्र' के बराबर और अंक 75 से ज्यादा या बराबर हो

Hindi Query: उन सभी विद्यार्थीयो का नाम ,अंक बताओ जिनका शहर 'कुरुक्षेत्र' के बराबर और अंक 75 से ज्यादा या बराबर हो

SQL Query: select student.name,student.marks from student where city = 'कुरुक्षेत्र' and marks >= 75

2 record(s) found.

name	marks
कृपाल	76
लोकेश	77

Figure 3: Example query output

V. CONCLUSION

The proposed an improved Hindi Language graphical user Interface to database to STUDENT database that store information of students accept query in Hindi language and tokenize the input sentence. Type of command is getting by mapping the tokens with command_type table. Then tokens are mapped with tables which store tables name, columns name, condition, functions and operations to find the table name, column name, function and conditions and stored in string to use them in SQL formulation. Then all the stored tokens are mapped with their corresponding English word. Now SQL query is generated and executed. Output of query is displayed to user.

VI. REFERENCES

[1]. A. Faraj El-Mouadib, S. Zubi Zakaria, A. Ahmed Almagrous and S. Irdess El-Feghi "Generic Interactive Interface to Databases", International Journal of Computers issue 3, vol. 3, 2009.

[2]. M. E. Saleh, "Semantic Based Query in Relational Database Using Ontology", Canadian Journal on Data, Information and Knowledge Engineering, vol.2, 2011.

[3]. B.J. Grosz "TEAM: A Transportable Natural-Language Interface System", In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pp. 39–45, 1983.

[4]. B.J. Grosz, D.E. Appelt, P.A. Martin, and F.C.N. Pereira, "TEAM: An Experiment in the Design of Transportable Natural Language Interfaces", Artificial Intelligence, vol.32, pp. 173–243, 1987.

[5]. A. Shingala, P. Virparia ," Enriching Document Features for Effective Information Retrieval using Natural Language Query Interface", International Journal of IT, Engineering and Applied Science Research, ISSN: 2319-4413, 2012.

[6]. A. Shingala, P. Virparia, “Enhancing the Relevance of Information Retrieval by Querying the Database in Natural form”, International Conference on Intelligent Systems and Signal Processing (ISSP), 2013.

[7]. Yuk Wah Wong, Learning for Semantic Parsing Using Statistical Machine Translation Techniques, Technical Report UT-AI-05-323,University of Texas at Austin, Artificial Intelligence Lab, October 2005.