Volume: 3 Issue: V Month of publication: May 2015

DOI:

www.ijraset.com Volume 3 Issue V, May 2015
IC Value: 13.98 ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Interface to Database By Using English Language

Amol Jadhav[#1], Sanket Bhuskat[*2], Sushant Raut[#3] Umesh Sutar[*4]
[#]Department of Computer Engg., S. P. University, Pune

***Abstract*— Information is playing an important role in our lives. One of the major sources of information is databases. Databases and database technology are having major impact on the growing use of computers. Almost all IT applications are storing and retrieving information from databases. Retrieving information database requires knowledge of database languages like SQL. The idea of using Natural Language instead of SQL has prompted the development of new type of processing called Natural language Interface to Database. NLIDB is a step towards the development of intelligent database systems (IDBS) to enhance the users in performing flexible querying in databases. This system prepares an "expert system" implemented in prolog which it can identify synonymous words in any language. It first parses the input sentences, and then the natural language expressions are transformed to SQL language. This paper is an introduction to Intelligent Database System and Natural Language Interface to Databases.**

***Key Words*— SQL, NLDBI, IDBS, DBMS, NLP**

I. INTRODUCTION

Nowadays there are too many data which maintain in organizations, companies and universities databases, but only the individuals who are familiar with data query methods can directly use these data. It is clear that if people can ask their question in natural language then the desired data prepare, the process will continue faster and with higher quality. NLIDB (Natural Language Interface to Database) is a technique which can make the computer understand the languages naturally used by humans, but not by artificial or man-made language such as a programming language. The main aim of the topic is asking questions to databases in natural language which is a very convenient and easy method of data access, especially for casual users who do not understand complicated database query languages such as SQL.

To allow easy access to database. Nontechnical person can also access database without using SQL queries. Translate the text into another language. Answer questions about the contents of the text. Use of database is world-wide today so NLIDB can allow easy interface for users. Allows user to interact with database system without knowing database schema. To work with any RDBMS one should know the syntax of the commands of that particular database software (Microsoft SQL. Oracle, etc.), Here the natural language processing is done on English i.e. the input statements have to be in English. -Input from the user is taken in the form of questions (WH - words like what, who, where, etc).

A complete NLIDB system will benefit us in many ways. Anyone can gather information from the database by using such systems. Additionally, it may change our perception about the information in a database. Traditionally, people are used to working with a form; their expectations depend heavily on the capabilities of the form. NLIDB makes the entire approach more flexible, therefore will maximize the use of a database.

II. LITERATURE SURVEY

The very first attempts at NLP database interfaces are just as old as any other NLP research. In fact database NLP may be one of the most important successes in NLP since it began. Asking questions to databases in natural language is a very convenient and easy method of data access, especially for casual users who do not understand complicated database query languages such as SQL. The success in this area is partly because of the real-world benefits that can come from database NLP systems, and partly because NLP works very well in a single-database domain. Databases usually provide small enough domains that ambiguity problems in natural language can be resolved successfully.

Here are some examples of database NLP systems:

LUNAR (Woods, 1973) involved a system that answered questions about rock samples brought back from the moon. Two databases were used, the chemical analyses and the literature references. The program used an Augmented Transition Network (ATN) parser and Woods' Procedural Semantics. The system was informally demonstrated at the Second Annual Lunar Science Conference in 1971.

LIFER/LADDER was one of the first good database NLP systems. It was designed as a natural language interface to a database of information about US Navy ships. This system, as described in a paper by Hendrix (1978), used a semantic grammar to parse questions and query a distributed database. The LIFER/LADDER system could only support simple one-table queries or multiple table queries with easy join conditions.

III. PROBLEM STATEMENT

The simplest and most useful NLP system to integrate is essentially a database NLP system that could be used to input a new problem in English. When students encounter a new problem that they cannot understand, the student could input the question as English text into the NLDBI interface, and then NLDBI can help the student to solve it. We use NLDBI to help add new questions simply by typing in the English text.

So we implement a system to interface database system by using spoken human English language.

IV. SYSTEM DESCRIPTION

A brief description of the system is as follows:

Suppose we consider a database say SQL. Within this SQL database I have placed certain tables, which are properly normalized. Now if the user wishes to access the data from the table, he/she has to be technically proficient in the SQL language to make a query for the SQL database. Our system eliminates this part and enables the end user to access the tables in his/her language.

Let us take an example:

Suppose if we want to view information of a particular employee from EMP table then we are supposed to use the following query:

SELECT * FROM EMP WHERE e_name =’ABC’;

But a person, who doesn’t know SQL, will not be able to access the database unless he/she knows the syntax and semantics of firing a query to the database. But using NLIDB, this task of accessing the database will be much simpler. So the above query will be rewritten using NLIDB as: Give the information of employee whose name is ABC. Both the SQL statement and NLP statement to access the EMP table would result in the same output the only difference being, a normal person who doesn’t know anything about SQL can easily access the SQL database.

A. The scope of the proposed system is as follows

1) To work with any RDBMS one should know the syntax of the commands of that particular database software (Microsoft SQL, Oracle, etc.).
2) Here the Natural language processing is done on English i.e. the input statements have to be in English. Input from the user is taken in the form of
questions (wh- form like what, who, where, etc).
3) A limited Data Dictionary is used where all possible words related to a particular system will be included. The Data Dictionary of the system must be regularly updated with words that are specific to the particular system.
4) Ambiguity among the words will be taken care of while processing the natural language.
5) All the names in the input natural language statement have to be in double quotes.
6) Data Dictionary used will be: - EMP, DEPT and PROJECT.

B. System Architecture

Generally NLIDB has following steps:-

1) Input English Statement
2) Grammar Check
3) Tokenization
4) SQL Query Generator
5) Result

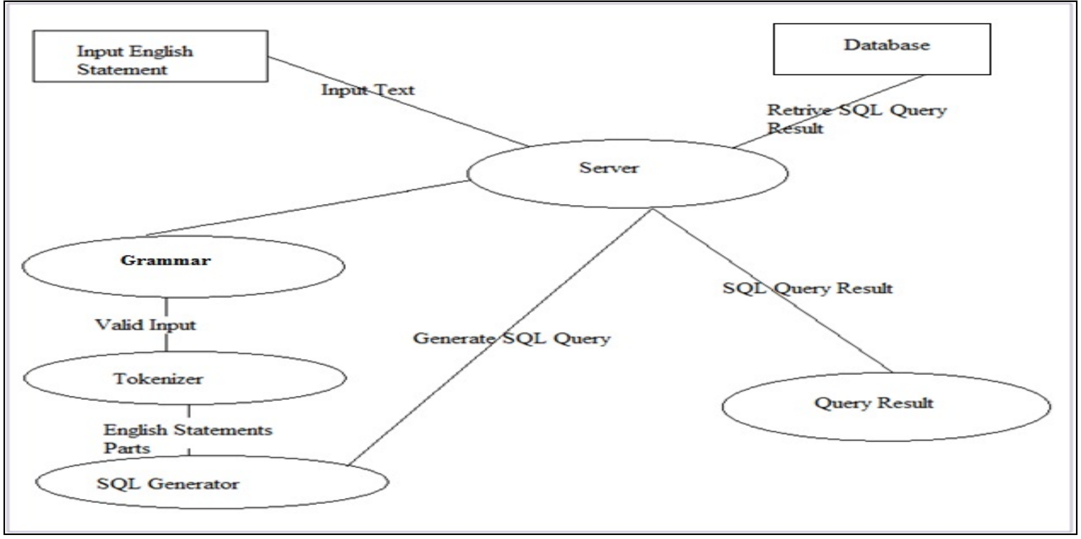

Fig-1. System Architecture

The system includes the following modules:

1) GUI: Designing the front end or the user interface where the user will enter the query in Natural Language.
2) Parsing: Derives the Semantics of the Natural Query given by the user and parses it in its technical form.

3) Query Generation: After the successful parsing of the statement given by the user, the system generates a query against the user statement in SQL and further gives it to the back end database.
4) Data Collection: This module collects the output of the SQL statement and places it in the User Interface Screen as a result form.

C. Semantic Grammar

There are two main parts of a semantic grammar. The first is a lexicon that stores all the possible words that the grammar is aware of. A simple entry in the lexicon might look like this:- (customer -> customer patron member) (customers -> customers patrons members) On the left-hand side of the '->' the word 'customer' defines a symbol that can be used in the grammar. When 'customer' is used in the grammar, it refers to the English words on the right-hand side of the '->' that is customer ',' patron or 'member'. Similarly, the plurals of these words are shown in the next line. Only single English words, or terminal symbols, can appear in the lexicon.

The other part of the semantic grammar involves rules to combine the terminal symbols in the lexicon to form phrases or sentences in a specific way.

For example, the rule (ATT_TAPE_CUSTOME -> RENTED by customer ATT_NUMBER)

Demonstrates how the TAPE.CUSTOMER attribute can be referred to in English. On the left-hand side is the non-terminal representation of the phrase. This can be used in other rules to refer to this phrase. On the right hand side is a combination of non-terminal symbols from other rules (in UPPER CASE) and from the lexicon (in lower case). RENTED represents another rule for synonyms of the verb 'rented', such as taken out, borrowed etc. Note that RENTED is not in the lexicon because 'taken out' is a two-word phrase, and so must be defined as a rule. The words 'by' and 'customer' appear in the lexicon, and the ATT_NUMBER symbol refers to a number attribute. Now the above rule can be used to parse the phrase "borrowed by member number 22", or "taken out by customer 14" etc.

D. Algorithm

1) Tokenization (scanning)
2) Split the Query in tokens
3) Give order number to each token identified
4) Split Query and extract patterns
5) Look for sentence connectors/criteria words
6) Break Query on the basis of connector/criteria tokens.
7) Use criteria tokens to specify condition in query.
8) Find attributes and values after criteria token

9) Map value for identified attribute and corresponding table
10) Replace synonyms with proper attribute names
11) Get intermediate form of Query
12) Transform it into SQL

E. Results

The user has to first login and then connect to the database. A database setting is required to access the database after getting the information about host name, database name, user name and the password. Once the database is setup, English Query can translate very complex English queries to SQL with the capability of searching multiple tables and multiple fields.
Steps followed to get the result:-

1) Type the natural language query in the dialogue box given.
2) Click on “Create English Statement” button.
3) The system will ask the user for the expected meaning. In case of ambiguities the user has to select the desired query.
4) Click on “Create SQL Statement” button.
5) The system will generate SQL query.
6) Click on “Run Query”.

F. Future Enhancement

More new grammar can be added to the parser to increase effectiveness. Adding a thesaurus is another suggestion, which could help automating the related words for table and column names. With the help of a thesaurus, the user input can be pre-processed to substitute related words with table or column names and also remove unwanted words. So far, this system considers selection and a few simple aggregations. The next step of the research is, to accommodate more complex queries.

V. CONCLUSION

In this system we translate the natural language statement such as WH-type questions statement into SQL query statement and for this we check the input statements are grammatically correct or not by using the grammar checking algorithm. If the statement is not a WH-type statement then it is invalid input and doesn’t work on our system.
We implement the tokenize, parsing, grammar checking algorithms for process the input, and system gives the output which data user wants. The advantage of NLDBI system is that it works on a Relational database and removes ambiguities.

VI. ACKNOWLEDGMENT

We wish to express our sincere thanks and deep gratitude towards Dr.D.V.Jadhav [Principal BSCOER] and our guide R.H.Kulkarni [DEAN Computer Dept.] for his guidance, valuable suggestions and constant encouragement in all phases. We are highly indebted to his help in solving my difficulties which came across whole Paper work. Finally we extend my sincere thanks to respected Prof. R.H.Kulkarni [Project Coordinator] and all the staff members for their kind support and encouragement for this paper.

REFERENCES

[1] Axita Shah, Dr. Jyoti Pareek, Hemal Patel, Namrata Panchal “NLKBIDB - Natural Language and Keyword Based Interface to Database”, AESICS, Ahmedabad University Ahmedabad, India IEEE 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)

[2] Siasar djahantighi F, Norouzifard M, Davarpanah S H,Shenassa M H. Using natural language processing in order tocreate SQL queries. In: IEEE International Conference on Computer and Communication Engineering (ICCCE); 13-15May 2008; Kuala Lumpur, Malaysia: IEEE. pp. 600 - 604.

[3] Li H, Shi Y. A Word Net-based natural language interface to relational databases. In: IEEE 2nd International Conference on Computer and Automation Engineering (ICCAE); 26-28 Feb.2010; Singapore: IEEE. pp. 514 – 518.

[4] Mrs. Neelu Nihalani, Dr. Sanjay Silakari, Dr. Mahesh Motwani. “Natural language Interface to Database using Semantic Matching", International Journal of Computer Application, Vol. 31, no.11, Oct. 2011 ISSN: 0975 – 8887.

[5] Owda M, Zuhair B, Crockett K. Conversation-Based Natural Language Interface to Relational Databases. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops; 5-12 Nov 2007; Silicon Valley: IEEE/WIC/ACM. pp. 363 – 367.

[6] D. S. M. Mrs.Neelu Nihalani, “A Natural Language Interface for Database:A brief review,” ijcsi International journal of computerscience, pp. 1694-0814, 2011.

[7] F. wang, “Parsing "Grammatically Incomplete Natural language queries to spatial databases,” in IEEE, Canada, 2001.

[8] en.wikipedia.org/wiki/natural_language_processing.