A generic model for Natural Language Interface to Database

B. Sujatha
Assistant Professor, Department of CSE
Osmania University
Hyderabad, *India*
sujatha_tha2003@yahoo.com

S. Viswanadha Raju
Professor of CSE
JNTUH College of Engineering
Jagtial, *India*
svraju.jntu@gmail.com

***Abstract*—Database management systems are used to store the data and to perform various operations on the data such as insertion, deletion and updation which is stored on the databases. To access the data from the databases, it is required to use a structured query language which is difficult to learn for the naive users. In this paper, designing of a generic natural language interface to the database management systems for retrieving the knowledge through structured query language is attempted. The proposed system can be used to retrieve the data from single columns and multiple columns. The system can also works on various types of queries which can join multiple tables with selected attributes with multiple conditions. The proposed system is independent of the underlying database tables and the relations among the tables. It is also independent of the natural language used to query the database.**

Index Terms*— *Natural Language Processing, Database Management Systems, Structured Query Language, First Order Logic

I. Introduction

Database systems are used since 1970s for the storing various kinds of data for different purposes such as commercial and personal needs. Though there are many types of architectures for database design like object oriented, object based, file based, hierarchical based and network based, the predominant designing of databases follow relational database architecture to store the data by using various types of storage devices. In relational databases, the data is stored using tables. The table contains set of rows and columns. Each column represent and attribute and each represents the instance of the data for a set of attributes. The data can be manipulated using various operators with fixed set of keywords by following a set syntax rules. By learning this structured query language one can extract the required data from the whole set of data, can also perform various operations such as update, manipulate and deletion of the data.

The Relational database management systems are more popular based on the characteristics like its robustness and flexibility, high performance, scalability, data security and protection and flexible data maintenance. Above all these advantages, it allows to index, perform aggregation, filtering and sorting can be done on the data using structured query language.

There are some disadvantages with relational databases. To perform operations on the data which is stored on databases, it is required to learn the structured query language. Hence , the naive user who knows only the natural language can not directly access the required information from the databases. To come out from these limitations, it is required to design a tool which can understand the requirements of the naive user through natural language query, convert the natural language query into an equivalent structured language query. Then the obtained structural query is used to access the required information from the databases. This kind of tool ins termed as Natural Language Interface to Databases or NLIDB system. Thus, the NLIDB system take the input as natural language query and converts it into a structures language query and returns the desired information to the naive user.

The designing of a NLIDB system for various languages and for different underlying databases is attempted by various researchers since five decades. But, designing of an most suitable NLIDB systems with high accuracy, precision and recall is still an open research problem which need to be addressed. The various earlier developed NLIDB systems focused on particular databases. There is need of designing a generic NLIDB system which can address the robustness and scalability of the applications. It is required to attempt the problem of portability to customize a NLIDB system to a other language and to other set of datasets designed for various domains. The efficiency of conventional NLIDB systems depend mostly on domain experts capabilities and linguistic features of the natural language.

In this paper, it is focused on designing a NLIDB system to overcome the various issues such as portability to different languages and to access the required information independent of the underlying database. It also required maintains the scalability and robustness of the system. To achieve this

978-1-4673-8286-1/16 $31.00 © 2016 IEEE
DOI 10.1109/IACC.2016.24

objective, the system is designed with general purpose syntactic parser. The parsed semantic frames of natural language its equivalent structured query is generated. In this paper, it is proposed a system which is successfully generated the semantics of inputed natural query and maintains a high accuracy 84% for student database.

II. RELATED WORK

There are many designing models are proposed in the literatures in the field of NLIDB such as pattern matching systems, syntax based systems, semantic based grammar systems and intermediate representation of languages system.

The pattern matching systems takes input as a set of rules and sample set of pattens. Based on the inputted word of sentence with natural language, it will be compared with the predefined patterns [1]. If there is a match between the input and predefined pattern then an action will be generated and these generated actions will be stored in the database. The response given to the user is based on the action generated. This kind of systems are limited to specific databases. The accuracy of the system is depend on the complexity of the patterns used to train and based on the set of rules used to train the system [2]. The NLIDB system SANVY is a good example for pattern- matching systems [3].

The syntax based systems takes the user query as input and parse the given input syntactically. The parse tree generated for the input query is overlapped with the one structured query of the database expressed using structured query language. LUNAR is a best example for syntax based NLIDB systems [4]. In these systems, the grammar rules are derived to match the various user questions with syntactic structures [5]. This system is used to answers the questions on rocks which were collected from the moon. With the corrections in the dictionary errors, the performance of the system has increased [8].

In the semantic grammar system, the parse is simplified by eliminating unimportant nodes or by combining two or more nodes into one node. The complexity of structured query can be reduced in semantic grammar system. Semantic grammar systems are more simpler when compared with syntax based systems. But these systems need to be trained with a prior knowledge of the various elements of a domain. PLANES and LADDER are the good examples for Semantic grammars systems [6,7].

In many NLIDB systems, the natural language query is transformed into an intermediate logical query. The logical query is represented using a meaningful representative language such as first logic language or Boyce codd normal form. This kind of representative languages, represents the meaning of the users queries in high order level of concepts. These concepts are independent from the structure of the database. This representative query is then transformed into an expression in the structured query language which can extract the relevant data from the databases.

In the intermediate representation of natural language systems, the natural language query is inputted to the system. This query is processed for syntax rules using a parser. Based on the set of syntax rules of a natural language, it generates a parse tree. By using the semantic rules of semantic interpreter module, the generated parse tree is translated into an intermediate logic query. In the semantics rule, left hand side of the syntax rule contains the logic expression of the constituent where as right-hand side of the syntax rule is a function of the logic expressions of the constituents. The logic expressions represents the words which are corresponds to lexicon. To get the required information from the database, the logic query is to be transformed into a structured query which is supported by the underlying Database Management System. MASQUE/SQL is an example of intermediate representation language systems [7].

By using semantic grammar techniques which interleaves semantic and syntactic processing in distributed databases, LADDER system is used to parse natural language questions to database understandable queries [7]. The another NLIDB system implemented using the language called Prolog was CHAT-80. This system transforms the natural language inputted English queries into Prolog expressions. These Prolog expressions are evaluated using the Prolog database. ROBOT which was a prototype of a NLIDB system named INTELLECT which was a commercial natural language interface to database systems [9]. ASK is the another NLIDB system which allows the users to train the system with new words and concepts while inter actioning with the system. By using the system, it is possible to make interactions with various external sources such as external databases, chating, Facebook, twitter, email programs and many other applications.

Generic Interactive Natural Language Interface to Databases (GINLIDB) was designed by the using UML and developed using Visual Basic.NET. The system was a generic system and it works for underlying suitable database and knowledge base [10]. SynTactic Analysis using Reversible Transformations (START) is also another Natural Language System. It was the first Web-based question answering system. It was available online and continuously operating till now [11]. It utilizes various language Dependant functions such as parsing, semantic analysis, word sense dis-ambiguous, natural language annotation for appropriate information segmentation and presentation for the user [12].

JUPITER was a NLIDB system to know the weather information worldwide. The user can pose a question to the system in their native language to forecast the weather information over the telephone. The Oracle Structured Query Language SQL can be learned by the students using the NLIDB system called SQL-Tutor. If the student asked the new

questions by typing at terminal then also, the SQL-Tutor can answer the question by using the existing knowledge [13]. KUQA system divides the query based on possible answer and after that it uses NLP techniques and also WorldNet to identify the answers which suitable to its corresponding category. But, this system can not handle any linguistic information [11]. QuALiM another NLIDB system designed based on complex syntactic structure which were based on certain syntactic description question patterns [11].

III. PROPOSED GENERIC MODEL FOR NLIDB

The progression of non-technical users in database community has led to the incorporation of natural language processing with database management systems. The main objective that grabs the attention is the development of a user friendly interface that efficiently processes user's database access request specified in a natural language statement. Natural language is always vulnerable to ambiguity. As a matter of fact mis communications arises among humans as well, so making a natural language interface to database such that it correctly interprets the user's request is not absolutely achievable.

The current system attempts to translate the database user's demand into a formal query language. The system observes the following architecture.

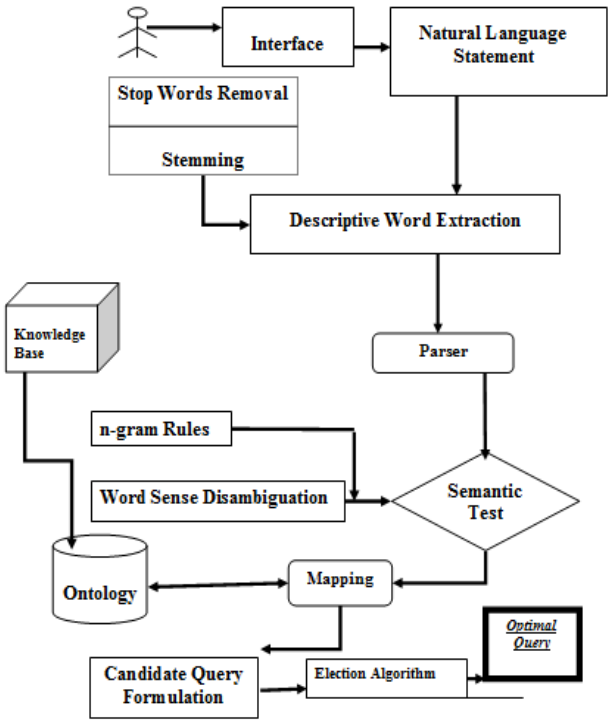

Fig 3.1 System Architecture

The natural language data access request issued by the user through the system's interface undergoes through the following processes for the final development of the SQL Query. They are Descriptive word Extraction, Parsing, Semantic Analysis, Candidate Query Formulation and Election Algorithm.

Descriptive word extraction presents the procedure of modeling the data in a way such that it processes the natural language statement to extract the words which are important for query generation. This is accomplished by making use of two well known processes from information retrieval system. They are Stemming and Stop-Word Removal.

Parsing is a vital stage of this development. The parser used is a Top- Down parser which analysis the given string symbols left to right. The top down parser is selected because the system aids in determining what to use rather than what to reduce which is the aim of bottom up parser. Parsing the given string is done using the logic called as First Order Logic (FOL). It is also known as predicate logic, used in mainly knowledge reasoning, a field of Artificial Intelligence.

The stereotype used for the conversion of natural language (NL) statement into SQL query is achieved by using a Rule-based approach, where the string matching features are also accomplished. According to the linguistic analysis of the given statement, the grammar defined is substituted to define the candidate query. The stop words removal and stemming procedures give the descriptive keywords necessary and sufficient to generate the query. The keywords are inspected as per the grammar rules defined below and are equated with the SQL select statement and its clauses.

To convert natural language query to first order logic conversion uses the notation to define the Backus- Naur Form BNF. There are many other techniques available to describe the language syntax such as Extended Backus Naur Form (EBNF), Augmented Backus Naur Form (ABNF) etc. BNF is a formal language to encrypt grammar for human utilization.

Each rule in BNF satisfies the format as Term: : =expansion. The basic formula to convert a natural language statement into First Order Logic is given by:

$$[<S1>] [\rightarrow] [<QEXP>]<S2>$$

S1 and S2 are the FOL statements that do not include any quantifiers or implication and QEXP is the Quantifier Expression. The symbol $\rightarrow$ specifies implications. The expression on the right hand side is in the Prenex Normal Form. It means that all the quantifiers of the right hand side are on the left side of the expression. A first order logic of the form,

$$V_1 w_1 \ldots . V_n w_n \, X,$$

where each of the V_i is a Quantifier having $\forall$ ("for all") or $\exists$ ("there exists") and X is a quantifier-free.

Semantic Analysis can be simply stated as figuring out the targeted meaning of a user's input. There exist relations within sentences that are to be apprehended to solve the ambiguous entities. Linguistic Analysis can be done using various processes like text recognition, document analysis, lexicography that is dictionary creation. Semantic analysis tries to recover the key meanings of words by extracting facts and relations among search objects in a sentence.

The ontology that is created for the conceptual use to have word sense disambiguation The polysemy for words has to be acknowledged and mapped with the database dictionary. Word sense disambiguation can be accomplished by observing the syntagmatic and pragmatic word associations. A syntagmatic relation can be defined as linguistic relations among words that appear successively.

EFFEN algorithm divides the query into into parts based on the occurrence of the preposition or verb phase obtained after preprocessing for POS tagging, the ontology built assists in mapping to the appropriate columns of the corresponding table and gives the response as the structured representation that is the subset of the database table. The splitting of query is done based on the place of occurrence of the verb and prepositions in the query in connection with the question words. The EFFCN system performance is measured in terms of retrieval efficacy using the information retrieval system metrics known as precision and recall.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. Database Description

The database consists of customer, product, vendor, invoice table for illustration. All of the tables are related to each other, pertaining to a relational database model. So, CPVbase can be defined as:

CPVbase = (Customer, Invoice, Product, Vendor). Customer table can be defined as

$C= \{ \forall n/ \ \exists c_k, w(n), x(n), y(n),z(n),\}$

where n represents a customer entity and c_k is unique to every, primary key of customer table and w, x, y and z are the functional variables of n, representing information related to n, as name, area code, phone and balance respectively. A product table is given as

$P= \{ \forall m/ \ \exists p_k, p_i(m),$ where $1 \leq i \leq s , v_k \}$

'm' is a product entity . p_k is the primary key and $p_i(n)$ denotes product's free variables, s represents the field size of the product. v_k illustrates foreign key from Vendor table. Vendor table can be defined as below

$V=\{ \forall d/ \ \exists v_k, v_i(d),$ where $1 \leq i \leq s \}$

'd' is a vendor entity. v_k is the primary key and $v_i(n)$ denotes vendor table's free variables, s represents the field size of the vendor. An entry into invoice table represents that a purchase transaction is at execution and is given by:

$I=\{ \forall e/ \ \exists I_k, I_i(e),$ where $1 \leq i \leq s , p_k , c_k \}$

'e' is an invoice entity . I_k is the primary key and $I_i(n)$ denotes Invoice table's free variables, s represents the field size of the Invoice. p_k , c_k are the foreign key references from the Product and the Customer table.

B. Evaluation measures

The attainment of relevant information by the user as per the natural language query in English gives the retrieval efficacy. The precision is the measure of retrieved results that are relevant to the need, evaluated using the fraction of relevant documents retrieved to the total number of documents retrieved. Mathematically it can be expressed as

$$Precision = \frac{CorrectlyAnsweredQueries}{AnsweredQueries} \quad (4.1)$$

Recall measures the relevant results retrieved as per the user statement. That is the fraction of relevant documents retrieved to the total number of relevant documents present in the system.

$$Recall = \frac{CorrectlyAnsweredQueries}{TotalNumberofQueries} \quad (4.2)$$

based on the precision and recall measurements, the system was tested for a random of 100 queries, giving the result tabulated displayed in table 4.1

Total Queries : 100
Answered Queries: 97
Unanswered Queries: 3
Correct Results: 84
Wrong Results: 13
Precision: 84/97 = 86.5%=0.86
Recall: 84/100 = 84.0%=0.84

Table 4.1 Implementation Results

The results shows that the system offers a recall rate of 0.84 which means that it has 84% Probability of generating correct responses to the user queries. This proves the effective and optimal working of the system. The result is determined by taking portion of queries and is obtained as presented in the table 4.2.

The table 4.2 contains system data generated by testing using different amount of queries and obtained the counts of correctly answered queries (correct_q), wrongly answered queries(wrong_g) and unanswered queries due to improper mapping or no corresponding record (unans_q) with their respective measures of precision and recall. The precision and recall is decreasing if irrelevant queries are observed. Thus the

precision and recall can be well defined if the data search is acquired with maximum relevant terms in the query.

No. of queries	Correct_q	Wrong_q	Unans_q	Precision	Recall
20	18	2	0	0.9	0.9
40	35	3	2	0.92	0.875
60	52	6	2	0.89	0.866
80	70	9	2	0.897	0.875
100	84	13	3	0.86	0.84

Table 4.2: Precision and Recall for varying number of Queries

V. CONCLUSIONS AND FUTURE SCOPE

The research aimed at developing an interface that eases the work of the naive user to formulate a database request and generate appropriate responses. The system vitally uses the ontology constructs, Parsing rules and FOL logic to extract the requisite information in forming a standard database Query. The system is flexible and can be adapted to any of the Database management systems or a relational database management system. EFFCN is a domain independent and highly portable system. It uses the semantics and syntactic knowledge to generate the correct match of the input statement's SQL query. Using the power of ontology and enhanced parsing mechanisms to filter query up to a refined level where it incorporates needed information as per the user. Compared to which the EFFCN system gives a success rate of 84% and high precision of 86.5%.

The NLIDB system future growth is directed towards improving the success rate by applying concepts of neural networks, machine learning parsing techniques and the use of SQL standard aggregate functions such as average, min and max along with the operator precedence concepts. The analysis of the system from the perspective of abbreviations and the temporal queries also needs careful interpretation along with the complex restrictions of FOL logic.

REFERENCES

[1] Mrs. Neelu Nihalani, Dr. Sanjay Silakari and Dr. Mahesh Motwani, "Natural Language Interface for Database: A Brief Review", IJCSI International Journal of Computer Science Issues, vol. 8, no. 2, pp. 600-608, Mar. 2011.

[2] T. Johnson, "Natural Language Computing-The Commercial Applications", The Knowledge Engineering Review, vol. 1, no. 3, pp. 11-23, 1984.

[3] Androutsopoulos, G.D. Ritchie and P. Thanisch, "Natural Language Interface to Databases-An Introduction", Department *of Computer* Science, University of Edinburgh, King"s Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland, U.K. , Mar. 1995.

[4] W.A. Woods, R.M. Kaplan and B.N. Webber, "The Lunar Sciences Natural Language Information System: Final Report", BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

[5] C.R. Perrault and B.J. Grosz, "Natural Language Interfaces", Exploring Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Mateo, California, 1988, pp. 133-172.

[6] D.L. Waltz, "An English Language Question Answering System for a Large Relational Database", Communications of the ACM, pp. 526-539, 1978.

[7] G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data", ACM Transactions on Database Systems, pp. 105-147, 1978.

[8] W. Woods, "An experimental parsing system for transition network grammars in Natural Language Processing", Algorithmic Press, New York, USA, 1973.

[9] L.R.Harris,"Experience with INTELLECT: Artificial Intelligence Technology Transfer", The AI Magazine, pp. 43-50, 1984.

[10] Faraj A. El-Mouadib, Zakaria S. Zubi, Ahmed A. Almagrous and Irdess S. El-Feghi, "Generic Interactive Natural Language Interface to Databases (GINLIDB)", International Journal of Computers, vol. 3, no. 3, 2009.

[11] "START Natural Language Question Answering System". [Online].Available: http://start.csail.mit.edu/

[12] M. Joshi, R. A. Akerkar, "Algorithms to improve performance of Natural Language Interface", International Journal of Computer Science & Applications, vol. 5, no. 2, pp. 52-68, 2008.

[13] Seymour Knowles and Tanja Mitrovic, "A Natural Language nterfacIe For SQL-Tutor", Nov. 5, 1999.