An Efficient Hindi Language Interface using Relational Databases

Priyanka Arora* ,Puneet Goswami
C.S.E Deptt, GGITC, Ambala, India
Priyanka5arora89@gmail.com, goswamipuneet@gmail.com

Abstract:

Database Management Systems have been used extensively for accessing, storing and retrieving data. However, database systems are not understandable to every user because they are hard to use. A plethora of e-governance applications railways, billings, agriculture, banks etc. use databases. Some users face difficulty in using these database systems because they do not have knowledge of the languages used in this system. So, they want a system that accepts a Hindi sentence as a query and after processing it, execute it and provides the output in the same language. Then the users have no need to learn any low-level languages those are hard to learn, use in databases such as SQL.

Keywords: HLIRD (Hindi Language Interface using Relational Database), NLIDB (Natural Language Interface to Database), SQL (Structured Query Language), DBMS (Database Management System), NL (Natural Language).

I. INTRODUCTION

As the requirement of information is essential part of our life. There are numerous sources of information, but the major one is database. Database helps us to store, access and retrieve information. No organization or industry is possible without the use of database. Each and every computer based application need to access information from database that requires knowledge of formal query language like SQL. But it is not possible for everyone to learn or write SQL queries. To overturn this problem many researchers have brought out to use Natural Language (NL) i.e. Punjabi, Arabic, English, Bengali etc. in place of formal query language which can be a perfect interface between an application of computer and non-technical user. This idea of using NL has induced the development of new sort of processing method in database systems. This new system can be named as Natural Language Interface to Database Systems (NLIDBs). It is a type of communication channel between the user and the computer. The user no longer needs to learn any SQL queries. Without the knowledge of any programming language, a user can act as a programmer. No such hectic queries are required from the user, by the system. It becomes very easy for a person to access data from database who has no knowledge of formal query language.

II. HISTORY

Researchers have made many developments in the area of natural language interface. Starting from the year 1970, the research work in this area is still in progress. Various different techniques have been proposed since then. The first system was introduced in 1971 i.e. LUNAR. This is a question answering system which answered the questions about samples of rock brought back from the moon [1]. The performance of LUNAR was impressive; it managed to handle 90% of requests without any error[3]. Then comes LADDER[2] (language Access to Distributed Data with Error Recovery). Basically, this system was designed for US navy ships with a natural language interface. It was designed for users having English as their natural language because it takes input in English language. This system was designed as a management aid to navy decision makers. In 1977, a system was developed named Philiq. It was popularly known as Philips question answering system. It uses a syntactic parser which runs as a separate pass from semantic understanding passes[10]. PLANES was developed in late seventies at the university of Illions, coordinated science laboratory for programmed language based enquiry. In eighties, Chat-80 came into existence and became very famous. It was implemented in prolog. The database of Chat-80 consists of facts about 150 countries of the world and a small set of English lexicon that is enough for querying a database. In 1983, ASK[5] system was developed. This system has its own inbuilt database. It allowed end users to teach the system new words during the interaction. TEAM came into view in 1987. It was designed to be easily configurable by database administrators with no knowledge of NLIDBs[4]. DATALOG is a database query system based in cascade ATN grammar by providing separate representation schemes for linguistic information and application domain, general world information and application domain knowledge. DATALOG attains a high...
degree of extensibility and portability. JANUS had similar abilities to interface with multiple systems (database, graphic device, expert system)[7]. All these systems could participate in the evaluation of a natural language request without the user ever becoming aware of the heterogeneity of the overall system. PRECISE was developed in 2004. The database of the system was in the form of a relational database and used SQL as a query language[8]. NALIX (NAtural Language Interface for an XML database). The database used for this system is XML (eXtensible Markup Language) with schema-free XQuery as the database query language. The basic idea is to use keyword search for databases. WASP (Word Alignment based Semantic Parsing) was developed to address the broader goal of constructing “a complete, formal, symbolic, meaningful representation of a natural language sentence”, it can also be applicable to the NLIDB domain[5]. A system named GINLIDB (Generic Interactive Natural Language Interface to Database) was developed in 2009. It was designed by the use of UML and developed using visual basic.net 2005. Along with these, there are a lot of other systems developed for variety of languages and for different purposes.

III. COMPONENTS of HLIRD
There are two components in HLIRD- Linguistic and Database Component. The task of Linguistic component is to handle the queries typed by the user in Hindi language, convert it into formal query, execute it and generate the output in Hindi language. Database component is responsible for performing all the database functions. Its main task is to execute the query and provide the result to Linguistic component.

IV. ARCHITECTURE of HLIRD
There are four phases in the Architecture of HLIRD system. The four phase are- Token filters, Parser, Query creator and DBMS. Output generated by upper phase is used by lower phase as it’s Input. Descriptions of all phases are given below:

A. Token Filters
This phase takes input from the user in Hindi language. It then splits the sentence of Hindi Language into tokens. All the tokens must be separated by comma, space, etc. from each other. These tokens are then stored in an array. Tokens may represent name of a table, column, row, command, operation, or it may be any value or any non-useful word. These all the tokens are output of this phase. Some tokens may be conditions or values etc. The output of this phase is all the tokens of sentence given by user in Hindi. If the user gives the query as:

“उन दिनों का शहर बताओ जिनका नाम 'नारस' या जन '01-08-1987' हो”

There are 12 tokens in this sentence. The token ‘वि’ is a name of table in the above sentence. Some of the tokens in this sentence are column names such as शहर, नाम and जन . Others may be values or conditions. This phase’s output is given to the next phase.

B. Parser
This phase is most important and takes the input from the Token Filter phase. All the Hindi tokens, their corresponding English word and token type are stored in lexicon or dictionary. These tokens may be name of a column, table or it may be any value, operation, command or something else. The tokens extracted by the upper phase are then matched one by one with the tokens stored in lexicon. If the match found then its corresponding English word with its type are saved. The tokens which are found useless are discarded in this phase. Only the useful Hindi tokens are kept. Now, we have with name of the table, attribute, conditions, and functions etc. that are used further to frame SQL query. This phase also contains sub phases which will be discussed in next section. The output of Token Filter phase is its input and it converts the tokens (that it takes from the above phase) into SQL query. Only those Hindi tokens are stored in lexicon that is enough to create an SQL query.
For every token of Hindi sentence, its corresponding English word and its type is also stored. The token stored may be a table name, column, function, condition etc.

A. Condition_start
The structure of Hindi sentence is considered first and assumed that some words in Hindi language always come before the beginning of a condition. The words in a sentence coming before the condition are stored in condition_start. Condition_start tokens are searched by the system in the array. When matching is correct then all the tokens are stored up to the sentence_end token except condition_start token in another array. There is a sentence_end token in the end of a sentence. Rest of the tokens of input sentence are stored into another array. For better understanding let we have an example:

उन सभी विख्यात के नाम बताओ जिनका अंक 70 से ज्यादा और अंक 80 से कम है

Here are Condition_start token is जिनका. All the tokens before and ahead it are stored in two different arrays. So, उन सभी विख्यात के नाम बताओ will be stored in one array named selection and rest of the tokens in another array named conditional_part but token जिनका will be stored in condition_start token. After this, these arrays are solved to get column name, table name and other useful tokens from array.

I) Table name, column name: To find the name of the table and column from Hindi sentence, the tokens stored in the selection array are matched with the lexicon tokens. If the table name is found then it is converted into the corresponding English word given in lexicon and then stored in table_name string. If not, then it is found from the name of column, in which the table resides. Now mapping of column is done. There may be one or more column name or it may not even contain any column. Now these columns are also converted into their corresponding English word and then stored in column_name string. If no column is found in array then column_name ‘*’ is set as default. In our example नाम is column name. For this, it is also checked that whether these columns belong to same table or not. So, there may be a possibility that these columns are from different tables, so joining of these tables will be required. Searching for their common field is required to join these two tables. This is stored in a string and further used to evaluate the SQL query. There may have some functions like avg(), sum(), min(), etc. If found, they are stored in function string. Now command name in array is searched in same way as we done before.

In our example, the token बताओ matches with the token stored in lexicon and its type is command and its corresponding English word is ‘select’ i.e. stored in
command string. All the purposeless tokens are discarded. After this, we left with name of table and column, command and function. The pattern of this part of SQL query is:

**Command function column _name**

from table name

So the outcome of this phase will be:

**Select name from student**

2) **Conditional _part:** In our example the second part after token जिनका is the conditional part. Now this part is solved to find the conditions and other valuable information. Four arrays are needed to solve the condition that are columns, values, conditions and logical operators. First token of conditional_part is matched with field type tokens. Condition always have columns, so if matching is true, column is stored in column_arraylist after its conversion to English word. It is also checked whether column belong to same column or not. Now matching of item of conditional_part is done with cond_opt_start type tokens. These token always come before the conditional operation. If it matches, this token is removed from conditional_part and stored incondition_arraylist after its conversion into English. Now logical operator if any is searched in the remaining conditional_part. If found, it is matched with the token on the first place in array with values stored in logop table, if it matches, it is converted into corresponding English word and stored into logicalop_arraylist. Then this token is removed from the conditional_part. Remaining tokens are then stored in the values_arraylist. Now, from the values stored in these four arrays SQL query is formulated. As per our example, the array list will have following values:

- **Column_arraylist** - marks
- **Condition_arraylist** - >, <
- **Logicalop_arraylist** - and
- **Value_arraylist** - 70, 80

Outcome pattern of this phase is first value from column_arraylist followed by first value from condition_arraylist followed first value from value_arraylist then first value from logicalop_arraylist. After this, second value from all these four arrays and so on.

The SQL query of this part will be:

**Marks> 70 and marks< 80**

Hence, as per our example the final SQL query is:

**Select student_name from student where marks>70 and marks<8**-

VI. **RESULTS**

There is a user interface in this system where a user can type the query, another text box to display the SQL query for the given input and a table for displaying the result of SQL query after its execution. In query creator frame, there are all the necessary tokens that can be used to frame a query. The tokens are column name, table name, functions and some other keywords of Hindi language. These tokens are given to make the system fast, efficient and easy to use.

![Fig. 3 execution of query in user interface](image-url)

In the above figure a sentence is given by the user. The user can either type the sentence or can copy the words by pressing प्रश्न निर्माता button and then pasting it in the given text area by pressing the पेस button. Then the user has to press क button then the system will convert the given sentence into SQL query. Then it is executed by pressing the लिप्य button. After this, the result will be displayed in the given table below in Hindi language. The user can also save his query by pressing the प्रश्न जमा button so that the user need not to type this query again if needed in future. The system supports all type of queries. We can test our system on more than 100 queries and it will handle very easily and the results will be promising.

VII. **CONCLUSION AND FUTURE SCOPE**

HLIRD is a system which accepts queries in Hindi language. Query can be given by user either by typing or selecting and copying the tokens from the query creator frame. After taking the input from the user, the sentence is parsed and all the tokens are stored in an array. After this,
mapping of these tokens will be done to separate out table name, column name, conditions, command and values. Then these tokens are converted to their corresponding English word. SQL query is then formulated using these words. Then it is executed and its output is displayed to the user. The system supports functions like avg(), min(), max(), etc., join, and other basic queries such as selection, updating and deletion on database. In HLIRD, there is no need for user to learn any formal query language which is difficult to learn. A user can give query in his native language. The HLIRD system can be extended by adding more tables and other complex queries can also be used using other techniques of semantic and syntactic grammar. For more efficiency, it can be extended using speech recognition technique in which system will taken spoken query by the user and result will be displayed.

REFERENCES


