Natural Language Statement to SQL Query Translator

Pranali P. Chaudhari
Asst. Professor, I.T. Department
MIT Academy of Engineering
Alandi (D), Pune

ABSTRACT
A Database is an organized collection of data. It is created to store large amount of data and retrieve it efficiently in less amount of time. To retrieve data from the database Structured Query Language (SQL) is used. SQL has its own syntax of query. To retrieve correct data from the database this query should be written in proper or correct syntax. Thus users should have sufficient knowledge of SQL to retrieve data. In this paper a light weight technique of converting a natural language statement into equivalent SQL statement is highlighted which when executed on database provides us with the accurate results. The main advantage of this technique is the users which are unknown to the syntax of SQL can also use the system and retrieve data.

General Terms
Natural Language Processing

Keywords
Structured Query Language, Natural Language Statement.

1. INTRODUCTION
Databases are very powerful means of storing and retrieving large amounts of data quickly and efficiently. There are many different commercially available database management systems used around the world. However getting data out of these databases is not an easy task. A special database interaction language called SQL (Structured Query Language) is used to communicate with these databases. Using Natural Language to communicate between a database system and its human users has become increasingly important since database systems have become widespread and their accessibility to non-expert users is desirable, if not essential, to facilitate full use of the database system. Natural Language to SQL translator (NLS-to-SQL) is aimed at reducing this complexity of database querying. First it is necessary to use a language that is understood by anybody, whether an expert database programmer or person with no computer knowledge. The best-suited language for this purpose is the English language. This means NLS-to-SQL has to translate English or natural language queries into SQL before retrieving data from database. Keeping this in mind I come up with a technique that converts a natural language statement to its equivalent SQL statement.

To make "NLS-to-SQL Translator" more flexible a lightweight approach is used to convert the Natural Language input into its SQL equivalent. This Lightweight approach extracts certain keywords and indicators from the English query using the preprocessor and then using the post processor, generates the SQL statement. The another important feature of this system is it provides a method of updating the dictionary by which we can add new word or phrase that can be mapped further to its equivalent clause in SQL. The rest of the paper is organized as follows: section 2 focus on the motivation for developing this system. Section 3 provides the literature survey from various papers in this area. Section 4 and 5 describes the system architecture and implementation in detail. The results of the system are detailed in section 6 along with the types of statements that are successfully implemented. Section 7, summarizes the results of the study and draw conclusions and the potential future work in this area.

2. MOTIVATION
Now days almost every IT applications require a database for storing and retrieving huge amount of data. To retrieve this data is an easy task for a person who has a good knowledge of data retrieval language like SQL. But at the same time if the person using the application doesn’t know SQL then it becomes a tedious job. Thus there is a need of a system which takes an English language statement (ie: natural language) and converts it into an equivalent SQL statement which when executed on the database results in accurate data. There are various approaches available for natural Language Statement to SQL Translator.

3. RELATED WORK
A lot of work has been done in the area of natural language support to database. However almost all the work that has been done uses process of applying semantic and syntactic analysis to get an logical representation of the sentence followed by a conversion of the representation into a database query [3,7,9]. However all these approaches do require a detailed syntactic and semantic analysis of the sentence which is computationally expensive. Generic Interactive Natural Language Interface to Database (GINLIDB) [1] is developed which consists of two major components linguistic handling and SQL constructing. Linguistic handling concentrates on the grammatical structure where second generates the SQL statement. Systems for support to a temporal database [2] has been proposed in which a prototype approach is used but it is limited to single sentences, multiple sentence query is not been handled. A model for automatically translating question in natural language to SQL using DB metadata and lexical dependencies is proposed [4] with the relational algebra form also. Token based and Template based methods are some of the approaches that are used to extract the representations [10]. A theoretical framework for reliable NLIs which is the foundation for the fully implemented PRECISE NLI [6] is proposed for a broad class of semantically tractable natural language questions. An introduction to natural language interfaces to databases (NLIDBs) [8] is provided where NLIDB architectures, portability issues, restricted natural language input systems (including menu-based NLIDBs), and NLIDBs with reasoning capabilities are discussed.

SQ-HAL [5] is the powerful system that can translates different types of select queries which include retrieving of data from multiple tables with or without conditions. But it cannot generalize other words which can be optional and may be omitted while generating queries. Also the support for synonyms for table name and column name is not present.

4. LIGHT WEIGHT NLS TO SQL TRANSLATOR

In NLS-to-SQL, input is the natural language statement which is the requirement specified by the user in terms of questions, which is given to the system. Internal system is divided into preprocessor and postprocessor and input is given to the preprocessor. Furthermore preprocessor will decide the type of the query (Select/Delete), substitute numerals and comparison operators, remove apostrophe and replace it by corresponding construction and then elimination of useless words and recognizing the keyword (extracts the noun clause).The extracted noun clauses may be the table or column name. Then output from the preprocessor is given to the post processor as input in which it initially recognizes the items in the statement(table values), extracts the tablename also aggregate function. Now the actual query formation starts. The postprocessor then decomposes the statement based on the words where, whose, which, such that, etc. Then postprocessor decide the query template type. And using this template query is translated in to its SQL equivalent. Fig. 1 shows the detail system architecture.

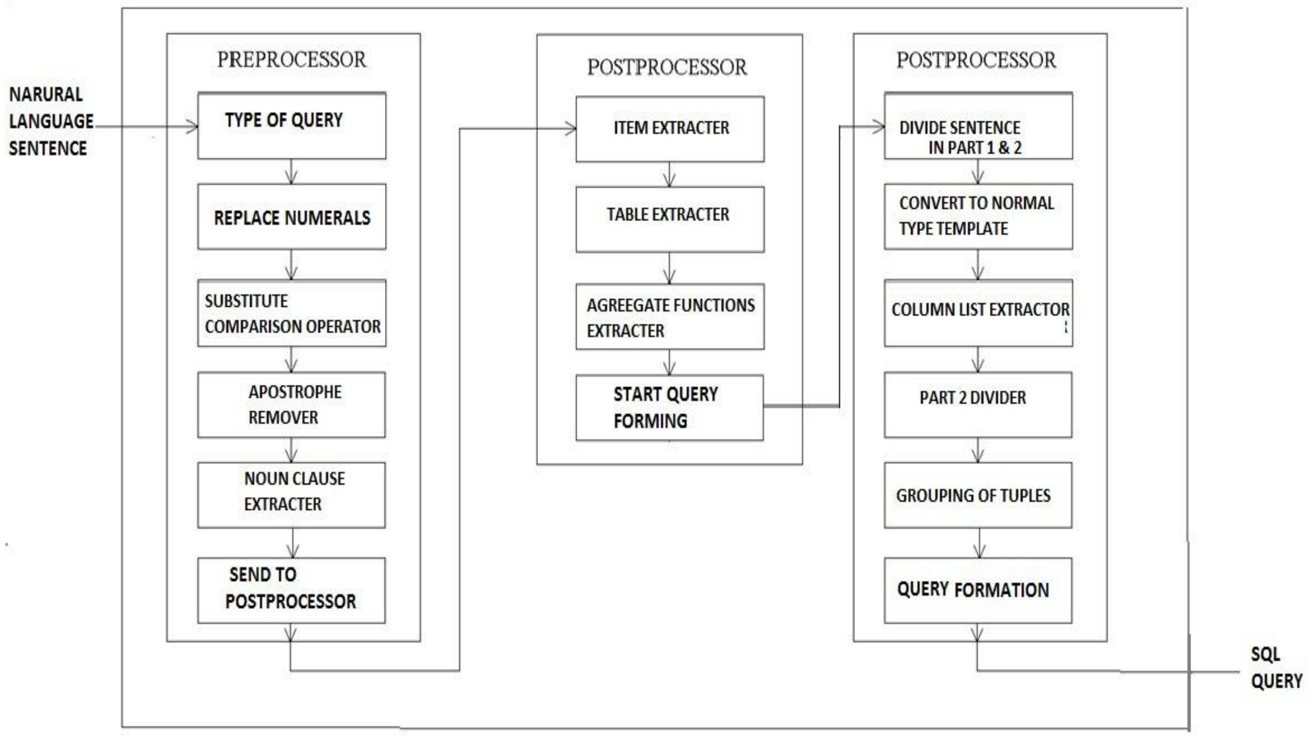


Fig. 1. System Architecture

5. IMPLEMENTATION

For implementation purpose the system is divided into three modules:

1. Pre-Processor
2. Post-Processor
3. Generation of SQL queries

5.1. Pre-Processor

Input to this module is an English sentence. Here first the type of query is identified and based on it preprocessor replace numerals, substitute comparison operators and also remove apostrophe. After that the noun clauses (ie: column or table name in the English sentence) are extracted. This output of preprocessor i.e english sentence containing noun clauses, substituted numerals and comparison operators is given to the postprocessor. For this, Convert Word to Number, Noun Clause Extractor , Numerical Operator Substituter interfaces are created.

5.2. Post-Processor

In this module, an interface named item extractor which extracts item (numerical value or table value), Table Extractor (identifies tablename), Sentence Divider (breaks the sentence as appearing keywords like where, whose, which, such that, etc.) in java.

5.3. Generation of SQL Query

This module, implements the interfaces created in the preprocessor and post processor and based on that SQL query is generated.
Fig.2 shows the data flow diagram for the system.

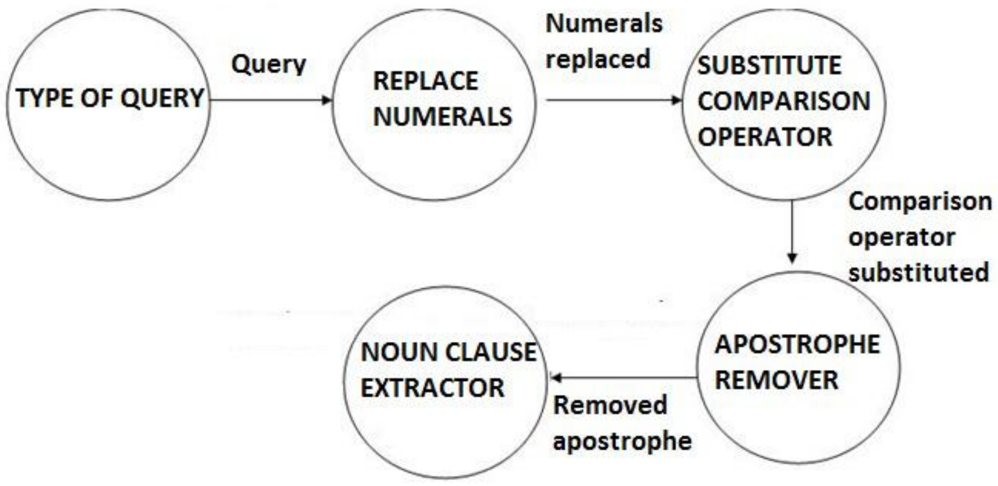
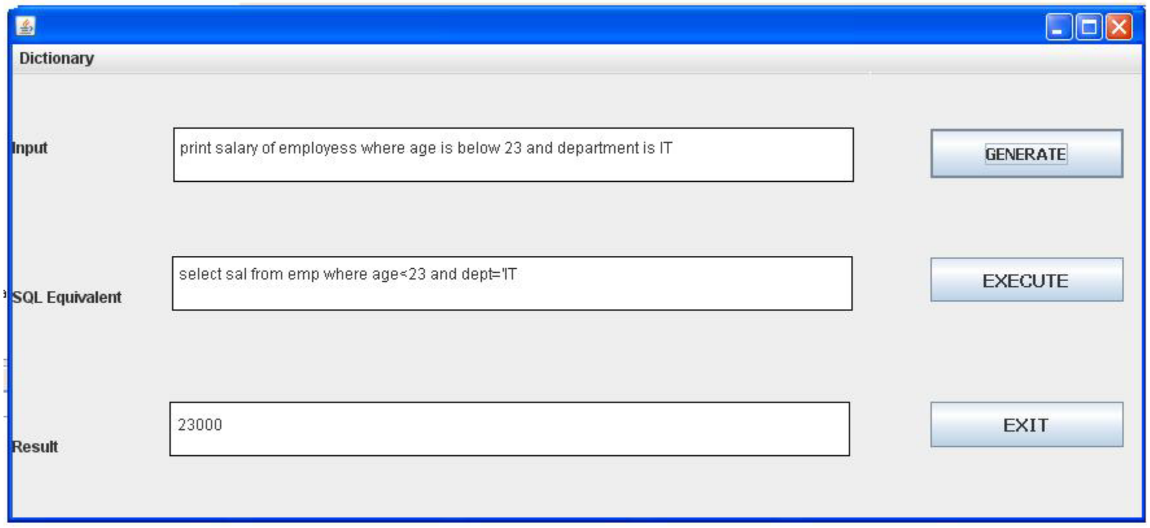


Fig. 2. Data Flow Diagram for System

6. EXPERIMENTAL RESULTS

To evaluate this system a single schema named employee with the attributes empid, name, designation, address, salary, age, gender, department, contactno. Mainly the select clause is focused rather than other DML clauses. Also the system is capable of translating the queries consisting of two conditions, having Apostrophe, aggregate functions etc. also there is a provision of converting numbers into words and vice versa (ie: 123 into one hundred twenty three and vice versa.). here a provision of updating a dictionary is also provided for the synonyms of words that are not being considered. If such a word comes in the natural sentence then the dictionary can be updated accordingly and then generate SQL query and execute it. Fig. 3 & 4 shows some snapshots of the system.



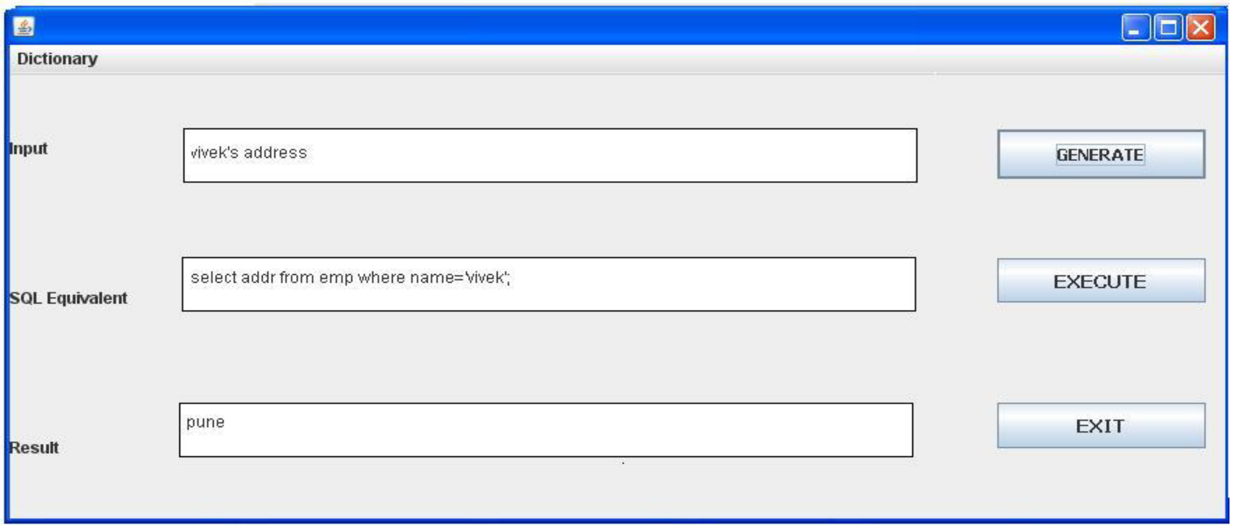
Input: print salary of employess where age is below 23 and department is IT

SQL Equivalent: select sal from emp where age<23 and dept='IT

Result: 23000

Fig. 3. Executing statement with two conditions

It is statement with two conditions. Here after substituting the noun clauses (ie: salary , age, dept) and extracting items (ie: 23, IT) also recognizing comparison operator (ie: below), identifying the table name with the help of them, forms the equivalent SQL query.



vivek's address

select addr from emp where name='vivek';

pune

Fig. 4. Executing statement containing apostrophe and without table name

Figure 4 shows the statement which contain apostrophe .Here the item with apostrophe (ie: vivek's) is substituted by “of vivek" and put it after the immediate noun clause (ie: address). So the original English statement now becomes “address of vivek" i.e normal english statement without any condition and table name.

Following is the list of statements that are successfully executed by the system:

Table 1. Working Statements

Example of Statement	**Comments**
select dept from emp	support for abbreviation
show name of employees	other synonyms of 'select' and 'from'
names of employees	no explicit 'select'
names, age and sex of employees	Multiple columns to be displayed column_list
employee names	support for missing 'from'
show id of employee with name is raj	support for 'where' and 'comparison operators'
show id with name raj	missing table name and comparison operator
salary of raj	missing table name
raj's sal	support for missing comparison column name
show name with sal *thirty thousand*	automatic number conversion
show name with sal greater than 30000	support for comparison ops in english
show name where sal is below 40 thousand and above 20000	support for 'above' and 'below'
show where sal > 30000 and add is pune	support for multiple comparison columns
name whose sal is between 30000 and 40000	support for between
name whose sal is between 40000 and 30000	inverted range
salary where name is karan, vivek or sagar	Multiple items mapped to same comparison column
select where *karan, vivek are names* and *30000 is salary*	inverted column_list order
select details where vivek and karan are names and address is pune and 30000 is salary	support for complex statements with 'are' and mixed order column_list
show highest sal	support for aggregate functions
show *youngest* employee	support for ambigious aggregate functions

7. CONCLUSION AND FUTURE WORK

A flexible Lightweight NLS to SQL translator is proposed in this paper. The system is capable of translating almost all English statements in an efficient way. Results show the implementation of aggregate functions and combination of two or more operations has been executed successfully. The important aspect of this system is functionality of updating the dictionary for the synonyms of the words. The system can be further enhanced to support multiple tables for the formation of SQL query. Also some mechanism of query optimization can also be used for better performance.

8. REFERRENCES

[1] Faraj A. El-Mouadib, Zakaria Suliman Zubi, Ahmd A. Almagrous, "Generic interactive natural language interface to databases (GINLIDB)", EC'09 Proceedings of the 10th WSEAS international conference on evolutionary computing, ISBN: 978-960-474-067-3

[2] Rani Nelken, Nissim Francez, "Querying temporal databases using controlled natural language", Proceeding COLING '00 Proceedings of the 18th conference on Computational linguistics - Volume 2

[3] Alessandra Giordani, Alessandro Moschitti, "Semantic mapping between natural language questions and SQL queries via syntactic pairing", Proceeding NLDB'09 Proceedings of the 14th international conference on Applications of Natural Language to Information Systems, ISBN:3-642-12549-2 978-3-642-12549-2

[4] Alessandra Giordani and Alessandro Moschitti, "Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked", Proceedings of COLING 2012: Posters, pages 401–410, COLING 2012, Mumbai, December 2012.

[5] SQ-HAL: Natural language to SQL Translator, http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura/

[6] Popescu, A.M., A Etzioni, O., A Kautz, H.: Towards a theory of natural language interfaces to databases. In: Proceedings of the 2003 International Conference on Intelligent User Interfaces, Miami, Association for Computational Linguistics (2003).

[7] Frank S.C. Tseng, Chun-Ling Chen, "Extending the UML concepts to transform natural language queries with fuzzy semantics into SQL" Information and Software Technology, Volume 48, Issue 9, September 2006, Pages 901–914

[8] Thanisch P Androutsopoulos I, Ritchie G. 1995.Natural language interfaces to databases - an introduction. Journal of Language Engineering.

[9] I-S. Kang, J-H. Bae, J-H. Lee, "Database Semantics Representation for Natural Language Access", CW '02 Proceedings of the First International Symposium on Cyber Worlds (CW'02), ISBN:0-7695-1862-1

[10] Desai B Stratica, N. 2004. "Schema-based natural language semantic mapping." In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems.