Volume 4, Issue 7, July 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Natural Language Interface to Databases-An Introduction

**Er. Amit Chaudhary**[1]
Assistant Professor, CSE,
Modern Institute Of Engineering & Technology,
Mohri, India

**Er. Annu Battan**[2]
Student, M.Tech-CSE,
Modern Institute Of Engineering & Technology,
Mohri, India

***Abstract— Database management system (DBMS) has been popular in all applications to store and manage data. It is very difficult for common user to access database using query languages as SQL. So a system is to be developed which make the user to access the database using their native language. This paper is an introduction to the natural language interface to databases (NLIDBS) and advantages and disadvantages of NLIDB. Some already developed systems are then discussed followed by the discussion of the components of NLIDB. The discussion then moves on to various techniques are discussed.***

***Keywords— NLIDB (Natural Language Interface to Databases), NL (Natural Language), DBMS (Database Management Systems), SQL (Structured Query Language), SMS (Short Message Services)***

## I. INTRODUCTION

It is difficult to access the database for persons having no knowledge of database language. In recent times there is an increasing demand for non-expert users to query relational database in a more natural language encompassing linguistic variables and terms. So, the idea of using natural language instead of SQL triggered the development of a new type of processing method i.e. Natural Language Interface to Database. Where users have no requirement to learn any other formal language, they can give query in their native language. Hence it discarded the burden to learn SQL. A NLIDB system will help us in many ways. By using such systems anyone can collect information from the database. Furthermore, it may change our thinking about the information in a database. Earlier, people are used to working with a form; their expectations depend heavily on the capabilities of the form. Natural Language Interface to Database makes the whole approach more flexible. There are various applications that can take advantages of NLIDB system. In PDA and cell phone environments, the screen is not as wide as of a computer. Filling a form that has many fields can be cumbersome as one may have to navigate through the screen, to look up the scroll box values, to scroll etc. Instead, with NLIDB, only work that needs to be done is to type the question similar to the SMS.

## II. ADVANTAGES OF NLIDB

Like other systems, NLIDB also have some merits as well as demerits. This section discusses NLIDB.s advantages over formal query language and form based interfaces [2]. Advantages are following:

- **No need to learn artificial language**

The user is not required to learn an artificial communication language. It is difficult to learn Formal query languages and master at least by non-computer specialists. In NLIDB user use their native language to query the database so there would be no need for the users to spend time learning the system communication language.

- **No need To know Physical structure of data is not required**

To query in formal language one should have knowledge of location of data where it is store. But there is no requirement of that data in NLIDB.

- **Easy to use**

To retrieve data NLIDB system require a single input while a form based may contain multiple input depending upon the capability of form. In case of query language a question may need to be stated using multiple statements which may contain one or more sub queries with some joint operations.

- **Discourse**

Another advantage of NLIDB invention in concern natural language interface that support anaphoric and elliptical expression. NLIDB of this kind allow the use of brief underspecified questions where the meaning of each question is accompanied by the discourse context.

*© 2014, IJARCSSE All Rights Reserved* 

## III. DISADVANTAGES OF NLIDB

Disadvantages of NLIDB [2] have been discussed below:

- **Dealing with limited set of natural language**

The linguistic capabilities of NLIDBs are not clear for the user. Current NLIDB can only cope with limited subset of natural languages. Users find it difficult to understand what kind of question the NLIDB can or can't cope with. For example MASQUE is able to understand ¡°what are the capitals of the countries bordering the Baltic and bordering swedan ?¡± which leads the user to assume the system can handle all kind of conjunctions. However the above question can't be handled, which infect can be answered. So the type of queries which the NLIDB support is not clear to the users, make it difficult to use the system.

- **Difficult to decide failure of query**

When NLIDB can't understand a question, it is generally not clear to the user whether the rejected question is out of system's conceptual coverage or it is outside the system's linguistic coverage. Thus user often try to give input by changing the phrase question referring to concepts the system does not know because they think the problem is caused by the system's limited linguistic coverage.

On other hand, user doesn't try to rephrase the question that system could conceptually handle, because they don't realize that the particular phrasing of question is outside the linguistic coverage and that different phrasing of same question could be answered. Therefore to decide whether the rejection was due to linguistic failure or it was because of conceptual failure is difficult. Some NLIDB attempt to solve this problem by giving diagnostic messages, showing the reason of a question can't be answered (e.g. Syntax too complex, unknown concept, unknown word etc.).

- **Ambiguity**

Natural language is claimed to be too ambiguous for human-computer communication. NLIDB user have to type long question, while in form based interfaces only fields have to be filled and in graphical interfaces most of the work can be done by mouse clicking and formal queries are more specific. Natural language question are often ambiguous. This may leave user with multiple answer of same question.

- **Wrong assumption by users**

NLIDB users are often deluded by the system's ability to process natural language. They assume that the system is intelligent so it can comprehend facts. While most of the NLIDBs have no reasoning capability. This problem doesn't stand up in graphical interfaces and formal query languages where the capabilities of the system are more clear to the user.

## IV. Applications of Hindi Language Graphical User Interface to Database Management System

Hindi is mostly spoken in northern and central India, Pakistan, Mauritius and Suriname. Approximately Seven hundred Million people speak Hindi as either the first or second language. Large numbers of applications uses database. For these people a system should be developed where they can access the database with Hindi language. We identified some areas where interface to database using Hindi language can be applied.

- **Agriculture**

In India two third of whole population is dependent on agriculture. Farmers are generally not much literate. They face lot of problems regarding irrigation, use of pesticides, time to reap the crop etc. Government has developed many systems to help farmers solving their queries. Data which is related to their queries is stored in databases. As we said above farmers are not much literate so we can't expect a SQL query from them. So there should be a system which is friendly to farmers. Using Hindi as a query language will be a good idea for this. Therefore interface to database with Hindi language can be applied to achieve this goal.

- **Railways**

Railway is the cheapest and largely used by public for transport. Railway has a database regarding the arrival and departure time, frequencies of trains, journey fares, reservation and cancelation of reservation etc. In India most of the people speak Hindi so a system is required for railway database where passenger give query in Hindi language and result is also shown to them in Hindi. This will be helpful for them. This is where Hindi language interface to database management system is used.

- **Weather Forecasting**

To know the weather conditions may be required for some persons or organizations. Farmers are one of them because for all the activities from reaping to harvesting of crops depends heavily on weather. So, to provide queries for weather in Hindi is very important. This is done with interface to database with Hindi language. Presently there is a large amount of NLP-based research carried out for the development of such type of systems. One modern natural language system in English is Jupiter.

*© 2014, IJARCSSE All Rights Reserved* 

- **Sports**

Sports are played in nook and corner of India. Specially, in cricket huge data is stored in form of tables. One who want to know anything about the game, any match result or scorecard, they just need to type what they want in Hindi. The interface to database will provide them their result. Consequently here also interface to database is helpful.
Hindi language interface can provide its services in all the areas where user is not aware of database query language and he need to extract data from database. This can be any ordinary database. So interface to database is in Hindi language is useful for the people who are handy with Hindi.

## V. SUB COMPONENTS OF NLIDB

NLIDB has divided into two sub-components by computing scientists [5]. One is Linguistic and other is Database component. These are briefly discussed below.

**1 Linguistic Component:** It handle input as natural language , translate it into formal query and generate output in natural language from the result which come after execution of query.

**2 Database Component:** It performs traditional database management functions. A lexicon composed of a number of tables that store natural language words and their corresponding mapping to formal objects that are used to create a formal query.
These tables can have entries of relations name; attribute names, verbs, adverbs etc. questions entered in natural language translated into a statement with the help of parser which tokenize the input. Then a formal query is formed by mapping these tokens into lexicon tables, which is executed and the result in natural language is given to user.

## VI. HINDI LANGUAGE INTERFACE TO DATABASE (2014)

**Architecture of the system**

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

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

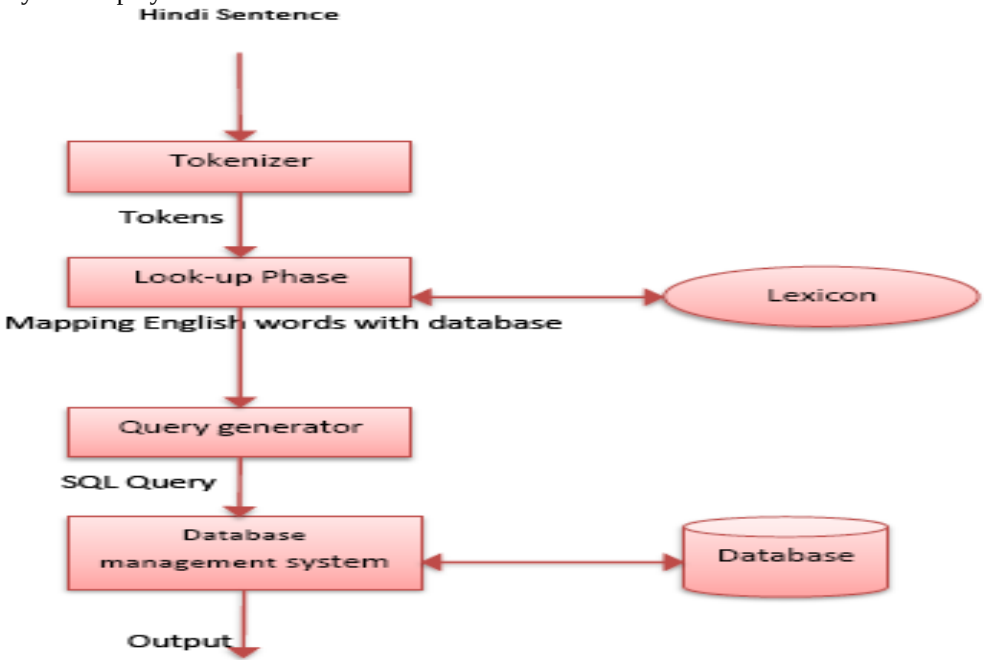


Figure: Architecture of the system

Detail of each phase is described below.

### 2.1 Tokenize input Hindi sentence

In this phase Hindi sentence is split into tokens. This is done with fact that all the tokens are separated by a space gap from each other. All the tokens which we get in this phase are stored in an array. Tokens are words of Hindi language. Token may be a table name, column name, condition, any value, command name, operation name or any non-useful word.
To understand this; let the user query is as:

*© 2014, IJARCSSE All Rights Reserved* 

उन सभी विधार्थीयो का नाम, अंक बताओ जिनका शहर 'कुरुक्षेत्र ' है

This Hindi sentence has 11 tokens. First token is उन .which is the starting of sentence. This is useless token which has no need in query formulation. Even in the absence of this type of tokens the query result will be unaffected. Likewise सभी is also non-useful token. Some tokens may be fields name as in the above query नाम and अंक are the field names. Input sentence also have token with table name विधार्थीयो. Therefore after this step we will have with all the tokens which the sentence is composed of.

**2.2 Semantic Parsing**

Lexicon store all the Hindi tokens, their corresponding English word and type of token whether it is table name, column name, any value, operation, command or something else. Tokens which we extracted in above step are matched with the tokens stored in lexicon one by one. If it matches then its corresponding English word is saved along with its type. This is the most important phase. All the useless tokens discarded in this phase only useful tokens are stored. After this step we will have with table name, field name (columns name), conditions, function etc. which will be further used to make SQL query.

**2.3 Formulate SQL query**

Now in the beginning of this phase we are with table name, column name, condition and command etc. SQL is generated in this phase according to Hindi sentence.

**2.4 Execute query and display result to user**

In this phase the above get SQL query is executed and result of which in Hindi language is displayed to user on the interface.

## VII. EXISTING NLIDB SYSTEMS

Since the end of 1960 there has been a large number of research work introducing the theories and implementations of NLIDBs. In natural language, asking question to databases is very convenient and easy method of data access especially for casually users who do not understand complex database query language. Prototype for NLIDB had appeared in late sixties and early seventies. Since then a number of systems have developed. Here we discuss some of them.

**3.1 LUNAR**

The system was introduced in 1971. It answers the questions about samples of rocks brought back from the moon [4]. 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. LUNAR system uses an augmented transition network (ATN) parser and procedural semantics. The performance of LUNAR was very impressive; it managed to handle 90% of requests without any error [5].

**3.2 LADDER**

The LADDER (Language Access to Distributed Data with Error Recovery) [6] 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. LADDER smears all the necessary information concerning the vocabulary and syntax of query, the name of specific attribute, how they are formulated and where the attributes are physically located to deliver the output. LADDER's first component is INALAND (Informal Natural Language Access to Navy Data), accept question in restricted subset of natural language and produce a query to the very large database (VLDB) as a whole. The questions to the VLDB as produced by INLAND refer to some specific attributes but no commitment about how the information in the database is dividing into files. For e.g. INLAND convert the question "what is the length of kenia ?" into the query ((NAM EQ JOHN # F.KENIA)(? LENGTH)). Where LENGTH is the name of the Length attribute, NAM the name of ship name field and JOHN # F.KENIA the value of the NAM field for the record concerned with kenia. Queries from INLAND are provided to the second component of LADDER called IDA (Intelligent Data Access). IDA split a query against the entire VLDA into a sequence of queries against individual files. IDA would comprise an answer that is related to the user's original query in addition to scheduling the correct sequence of file queries.

The third component of the LADDER system is for FAM (File Access Manager).The work of FAM is to find the location of the generic files and manage the access to them in distributed database. The system LADDER was instigated in LISP[15]. At the time of creation of LADDER system was able to process a database that is equivalent to relational database with 14 tables and 100 attributes.

**3.3 CHAT-80**

The system CHAT-80 [7] came in eighties and was implemented in prolog. According to [8] the database of CHAT-80 consist of facts about 150 countries of the world and a small set of English vocabulary that are enough for querying the database. The system translates the English language question by the creation of logical form as process of 3 serial and

*© 2014, IJARCSSE All Rights Reserved* 

complementary functions. First words are represented by logical constraints. In second function words, nouns and adjectives with their associated preposition are represented through predicates. Third function is representation of phrase through conjunction of predicates. The functions are following:

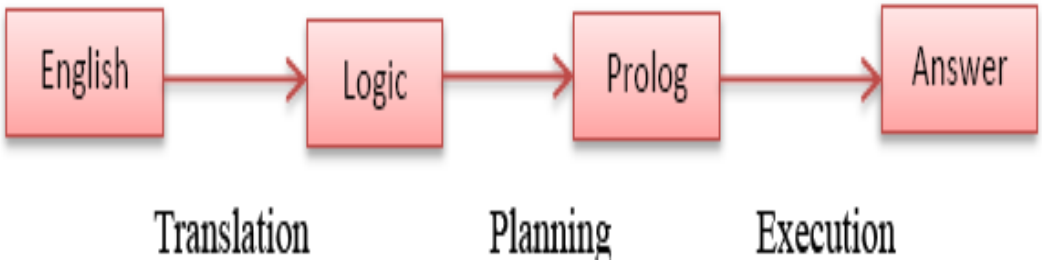


Figure: Working phases of CHAT-80[5]

**3.4 PLANES**

This system was developed in late seventies at the university of Illions coordinated science laboratory for programmed language-based enquiry laboratory. PLANES [11] include an English language front-end with the ability to understand and explicitly answer user request. This uses database of information of the US Navy 3-M (material, maintenance, management), it is database of aircraft maintenance and flight data. This system is mainly involved with problems of semantics and has three separate layers of semantic understanding. The layers are called "world model language", "English formal language", and "database language" and appear to correspond roughly to the external, conceptual and internal view of data.

**3.5 TEAM**

It was developed in 1987.TEAM was designed to be easily configurable by database administrators with no knowledge of NLIDBs [9][10].

**3.6 DATALOG**

DATALOG is a database query system based in cascade ATN grammar by providing separate representation schemes for linguistic information and application domain information, general world information and application domain knowledge. DATALOG attains a high degree of extendibility and portability.

**3.7 JANUS**

It had similar abilities to interface to multiple systems (databases, expert system, and graphics device) [12]. All the system could participate in the evaluation of a natural language request without the user ever becoming aware of the heterogeneity of the overall system.

**3.8 PPRECISE**

PRECISE is a system developed at the University of Washington by Alex Armanasu, Ana-Maria Popescu, Oren Etzioni, Alexander Yates and David Koin 2004.The database is in the form of a relational database using SQL as the query language. It presents the idea of semantically tractable sentences which are sentences that can be converted to a unique semantic interpretation by analyzing some lexicon and semantic constraints. PRECISE was examined on two database domains. The first one is the ATIS domain, which composed of spoken questions about air travel, their written forms, and their correct conversion in SQL query language. In ATIS domain, 95.8% of the questions were semantically tractable. The second area is the GEOQUERY domain. This contains information about U.S. Geography. 77.5% of the questions in GEOQUERY are semantically manageable. Using these questions, gives PRECISE 100% accuracy. The asset of PRECISE is based on the ability to match keywords in a sentence to the corresponding database structures. This process is completed in two stages, first by contracting the possibilities using Max-flow algorithm and second by analyzing the syntactic structure of a sentence. Therefore PRECISE is able to execute impressively in semantically tractable questions.

**3.9 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 [14].

The former component control the natural language query correctness as far as the grammatical structure and the successful transformation to SQL statement, opens a connection to database in use, execute the generated SQL statement and return the query result to the user. The user submitted query in English language is analyzed from the syntactic as well as semantic merits so the query will be correct and can be answered efficiently with respect to system knowledge base. Then the construction of valid SQL statement that represent the user's query is done and to retrieve the result of query to the user. The results of GINLIDB are very successful and satisfactory.

*© 2014, IJARCSSE All Rights Reserved* 

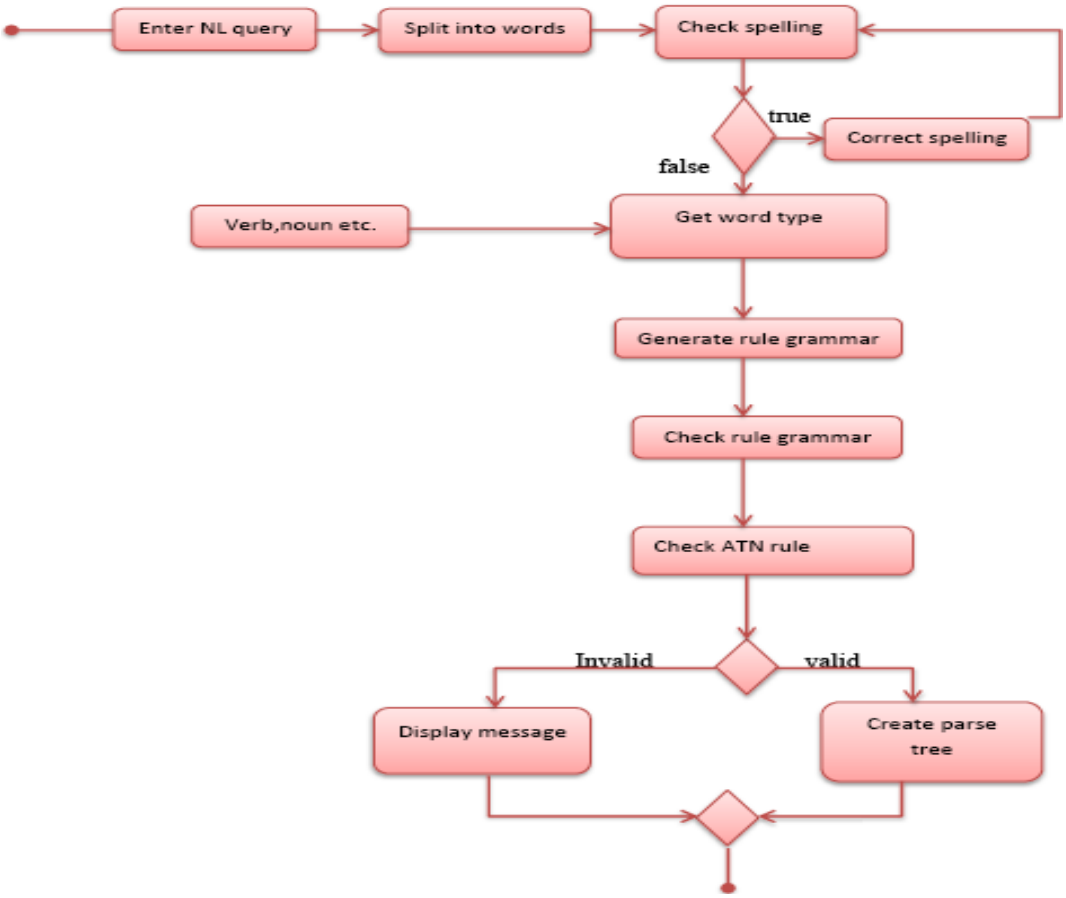


Figure: Activity diagram of GINLIDB [14]

**3.10 NALIX**

NALIX is Natural Language Interface for an XML Database. It is an NLIDB system developed at the University of Michigan, Ann Arbor by Huahai Yang, Yunyao Li and H. V. Jagadish in 2006. The database used for this system is XML (extensible markup language) with Schema- Free XQuery as the database query language. Schema-Free XQuery is a query language designed for retrieving information in XML. The basic idea is to use keyword search for databases. However, pure keyword search cannot be applied. Therefore, some ironic query mechanisms are added. Given a collection of keywords, each keyword has several XML elements to relate. All candidates are added to MQF (Meaningful Query Focus), which will automatically discover all the relationships between these elements. The main advantage of Schema-Free query is that it is not mandatory to map a query into the exact database schema, since it will automatically find all the relations given certain keywords. NALIX can be categorized into system that is syntax based; three steps are required for completing the transformation processes. First is generating a parse tree then validating the parse tree and third transforming the parse tree to an XQuery expression. NALIX is different from the other syntax based approaches; in the way the system is built. NALIX implements a reversed-engineering technique by building the system from a query language toward the sentences.

**3.11 WASP**

It stands for Word Alignment-based Semantic Parsing. It is a system developed at the University of Texas, Austin by Yuk Wah Wong [13]. 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 [13]. WASP learns to build a semantic parser given a corpus a set of natural language sentences annotated with their correct formal query languages. It has no prior knowledge requirement of the syntax, because the entire learning process is done using statistical machine translation techniques.

## VIII. TECHNIQUES USED FOR DEVELOPING NATURAL LANGUAGE INTERFACE TO DATABASE (NLIDB)

**4.1 Pattern-Matching Systems**

In the pattern matching based systems, the database specific were inter-mixed into the code, limited to specific databases, to the number and complexity of the patterns. The main advantage of the pattern-matching approach is its simplicity. In such systems no detailed parsing and interpretation modules are required, and the systems are also easy to implement.

*© 2014, IJARCSSE All Rights Reserved* 

**4.2 Syntax -Based Systems**

In syntax-based systems the users query is parsed and the resulting parse tree is directly mapped to an expression in database query language. Syntax-based systems manage a grammar that defines the possible syntactic structures of the user's questions. The main advantage of using syntax based approaches is that they deliver detailed information about the structure of a sentence. A parse tree includes a collection of information about the sentence structure; starting from a single word and its part of speech, how words can be clustered together to form a phrase, how phrases can be assembled together to form more complex phrases, until a complete sentence is built. By having this information, we can map the semantic meanings to certain production rules.

**4.3 Semantic Grammar Systems**

The basic idea of a semantic grammar system is to make simpler the parse tree as much as possible, by removing unnecessary nodes or combining some of the nodes simultaneously. Based on this idea, the semantic grammar system can better reflect the semantic representation without having complex parse tree structures. The main problem of semantic grammar approach is that it needs some prior knowledge of the elements in the domain, therefore making it hard to port to other areas. In addition, a parse tree in a semantic grammar system has specific structures and unique node labels, which could hardly be useful for other applications.

**4.4 Intermediate Representation Languages**

Intermediate Representation System was proposed because the difficulties of directly translating a sentence into general database query languages using syntax based approach. The idea is to translating a sentence into a logical query language first and then further converts this logical query language into a general database query language. In this process there can be more than one intermediate meaning representation language [1].

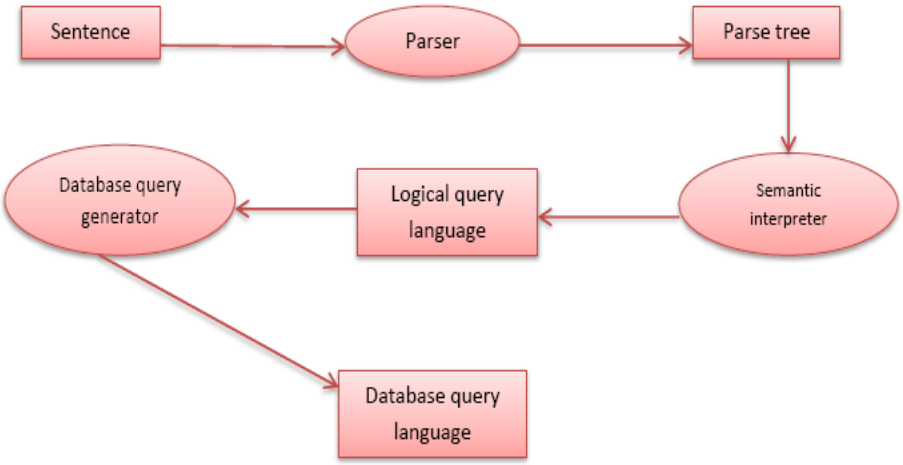


Figure: Intermediate Representation Language Architecture [1]

## IX. CONCLUSION

From the last few decades Natural Language Interface to Databases System has been developed. With the help of NLIDB user can interact with Database easily. But the use of NLIDB system is not wide-spread and not a primary choice for interfacing to Database. Reason behind this problem is large no. of deficiencies in NLIDB system in order to understand a NL (natural language). In the future, work could be done to improve the linguistic coverage by the NLIDB system. Use of NLIDB system could be made easier by avoiding the users to type long questions and allowing the system to display proper error message in case of failure of any query.

## REFERENCES

[1] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch, Natural Language Interfaces to Databases - AnIntroduction, Journal of Natural Language Engineering 1 Part 1 (1995), 29–81.

[2] Ana-Maria Popescu, Oren Etzioni, Henry Kautz, "Towards a Theory of Natural Language Interfaces to Database", University of Washington, Computer Science Seattle, WA 98195, USA

[3] Himani Jain, Mr. Parteek Batia " Hindii language interface to databases" Thapar university. Patiala – 147004 June [2011].

[4] 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

[5] N. Nihalani, S. Silakari, M. Motwani"Natural Language Interface for Database: A Brief Review", International Journal of Computer Science Issues, vol. 8, Issue 2, March 2011.

*© 2014, IJARCSSE All Rights Reserved* 

[6] 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.

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

[8] T. Amble, "BusTUC - A Natural Language Bus Route Oracle." 6th Applied Natural Language Processing Conference, Seattle, Washington, USA, 2000.

[9] 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.

[10] 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.

[11] R.J.H. Scha, "Philips Question Answering System PHILIQA1", In SIGART Newsletter, no. 61, ACM, New York, 1977.

[12] P. Resnik, "Access to Multiple Underlying Systems in JANUS", BBN report 7142, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1989.

[13] Priyanka Arora,"A Review on Nayural Language Interfaces To Relational Database" ISSN 2319-9725.

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

[15] Manika Tyagi," Natural Language Interface to Databases: A Survey", International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064, Impact Factor (2012): 3.358

*© 2014, IJARCSSE All Rights Reserved* 