Natural Language Interface to Database using Semantic Matching

Neelu Nihalani
Associate Professor
RGPV
Bhopal, MP India

Dr. Mahesh Motwani
Associate Professor
RGPV
Bhopal, MP India

Dr. Sanjay Silakari
Professor & Head CSE
RGPV
Bhopal, MP India

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. In order to retrieve information from a database, one needs to formulate a query in such way that the computer will understand and produce the desired output. The Structured Query Language (SQL) norms are been pursued in almost all languages for relational database systems. However, not everybody is able to write SQL queries as they may not be aware of the structure of the database. So there is a need for non-expert users to query relational databases in their natural language instead of working with the values of the attributes. The idea of using natural language instead of SQL, has promoted the development of Natural Language Interface to Database systems (NLIDB). The need of NLIDB is increasing day by day as more and more people access information through web browsers, PDA's and cell phones. In this paper we introduce an intelligent interface for database. We prove that our NLIDB is guaranteed to map a natural language query to the corresponding SQL query. We have tested our system on Northwind database and show that our NLIDB compares favourably with MS English Query product.

Keywords: Databases, Database Management System (DBMS), Structured Query Language (SQL), Natural Language Interface for Databases (NLIDB)

1. INTRODUCTION
Databases are the common entities that are processed by experts and non-experts with varied levels of knowledge. Databases respond only to standard SQL queries. It is highly impossible for a common person to be well versed in SQL querying as they may be unaware of the database structure namely tables , their corresponding fields and types , primary keys and more. NLIDBs were proposed as a solution to the problem of accessing information in a simple way, allowing ideally any type of users, mainly inexperienced ones, to retrieve information from a database (DB) using natural language (NL).On account of this we have designed an intelligent layer which accepts common user's imperative sentences as input and converts them into standard SQL queries to retrieve data from relational databases based on knowledge base. The primary advantage of NLIDB is that it conceals the inherent complexity involved in information retrieval based on unqualified user queries and it avoids the tedious process of configuring the NLIDB for a given domain.
Our Natural Language Interface to database system accepts queries in English language and attempts to understand them. Interface maintains its own dictionary. This dictionary contains words related to database and relationships. In addition to this, interface also maintains pre-defined data structures and in order to interpret the query. NLI refers to words in its own dictionary as well as to the words of standard dictionary. If the interface interprets the natural language query successfully, the corresponding SQL query is generated and submitted to the DBMS for processing.

2. MAIN OBJECTIVES
Before describing the characteristic of the system, it is necessary to discuss the context where our proposed solution is applicable and the background that took in account the problem**.**
We can access the databases in the following ways:

- We can access/retrieve data through application programs that are specially designed for the database.
- We can operate on data directly through relational language.

The second one is unavoidable, when an occasional operation (in particular terms) is performed. Our problem relates exactly to this situation, when human operator tries to directly interact with the database through relational languages, in order to retrieve/search data. This kind of interaction is often useful to those who are not specialists in informatics; they are interested only in looking up data. They may be managers or analysts, or individuals accessing the database.

3. BACKGROUND
According to Androutsopoulos et al. (1995), the earliest natural language interfaces to databases (NLIDB) research started in the late sixties. At that time, most of the research has concentrated on one database at a time as the implementation target, therefore they could not be modified to be implemented on other databases. One of the well-known NLIDB system in the sixties is Lunar [1], a system developed for a database which contained information about chemical analysis of moon rocks.

By the mid- eighties NLIDB was a very popular research topic, indicated by the numerous systems were being implemented [2]. During this time, the research focus had changed to the issue of portability, and some systems were even brought to commercial use, though they did not get the expected gain. The main reason for the lack of acceptance is probably due to the difficulties to fully understand a natural language.
The area of NLP research is still very experimental and the systems so far has been limited to very small domain[1]. When the systems are scaled up to cover large domains ,conversion of natural language to SQL becomes very difficult due to vast amount of information that needs to be incorporated in order to parse statement.
Despite the achievements attained in Natural Language Interfaces, present day NLIDBs do not guarantee correct translation of natural language queries into database languages. However there are many problems which have not been fully or adequately solved by existing NLIDBs. The main issue is of domain independence.

Some of the most important NLIDBs that are domain dependent VILIB [3], Kid [4], . PLANES [6]. In this type of interfaces the percentage of correctly answered queries is high (69.4–96.2% [5]), mainly because they are limited to one domain. In domain-independent interfaces the success percentage is usually lower than that of domain-dependent interfaces.

The most important of domain independent interfaces are: EnglishQuery [7,19], PRECISE [8], ELF [9], SQLHAL [10], and MASQUE/SQL [11]. Rendezvous [12], STEP [13], TAMIC [14], CoBase [15], CLARE [16], LOQUI [17], and Inbase [18].

Researchers have proposed numerous methods for mapping of natural language to SQL. The works developed on these NLIDBs are diverse. In these works different approaches for domain independent interfaces. We have proposed a different approach for domain independent interface and have compared with the commercially available natural language interface English Query[7]. Experiments were conducted to compare the performance of the interfaces on the basis of accuracy.

4. THE INTELLIGENT NLIDB SYSTEM ARCHITECTURE

Here, we present the architecture, of the Intelligent Natural Language Interface to Database system. The system is designed to accept any relational database schema. Our NLIDB system accepts users natural language sentences as input , parses them semantically and builds an SQL query for the database. The core functionality is based on the semantics and rules, which can be modified by the system administrator. Our system is composed of two modules : a pre-processor and a run time processor. Figure 1 shows the architecture of our NLIDB.

4.1. Pre-processor

The pre-processor automatically generates the domain dictionary by reading the schema of the database, uses WordNet to create semantic sets for each table and attribute name. The pre-processor also creates the rules that can be edited by the system administrator. Our system addresses the semantic parsing through the use of rules that are generated by the pre-processor. The rules are based on the schema of the database, which describe the relationship between the tables and their attributes. These are generated automatically when the system is configured for the given database and can be edited by the administrator.

4.2. Domain Dictionary

Domain dictionary which consists of metadata set and semantic set are generated automatically by the interface for the database. Data dictionary also contains rules related to the database schema which are derived from the Metadata set.

4.2.1. The Metadata Set

In general, a database is termed as set of tables organized in some common structure. The vital information that briefly describes the tables in the database is organized into a metadata set (M). The metadata set holds entries for all the '*n*' tables in the relational database with all their corresponding fields, foreign keys and their unique primary key.

4.2.2. The Semantic sets

In our system there are two types of semantic sets, first is the single lexicon semantic set and the second is a composite lexicon semantic set. The single lexicon semantic set consists of individual words and some of their synonyms that are used in English Language grammar. The composite lexicon semantic set is a combination of terminal words that form phrases or sentences in a specific structure. The pre-processor of the system uses WordNet to relate words semantically. Each word is assigned a family of synonyms and hyponyms, which forms the semantic set of the word. The semantic set (S) contains the list of all possible semantics related to table names and fields in the database. Semantic set for tables S_T and fields S_F are and respectively.

4.2.3. Rules related to database schema

The schema of the database gets translated into the rules. These rules are produced by the pre-processor, and they are based on the relationships between the tables. If in a query two tables Suppliers and Shippers are referenced and field Sno in Shippers table is a foreign key which references Sno field in Supplier table , then the attribute equation Suppliers.Sno = Shippers.Sno should be used with in the where clause and this rule is added in the table.

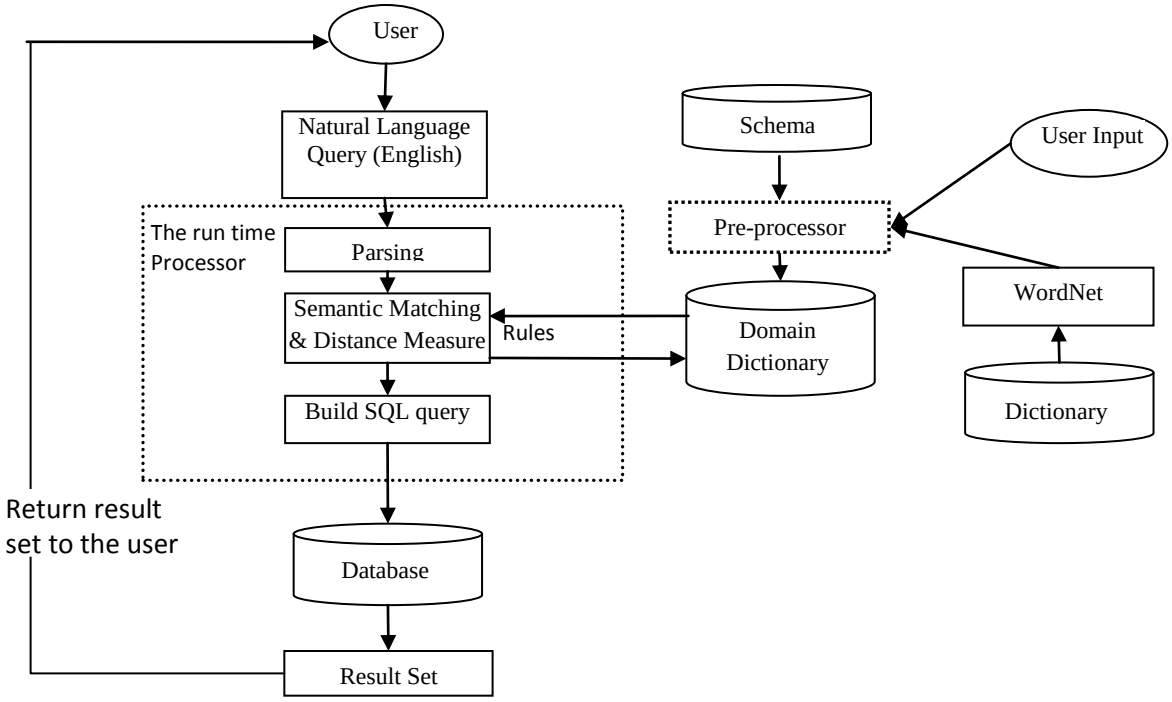

Figure 1 Architecture of our NLIDB

4.3. Run Time Processor

The run time processor uses the predefined data structures to parse the statements and tries to match the input words with the domain dictionary to extract the tables and fields involved in the natural language query.We have assumed that the tables and attributes names in the schema are meaningful and can be found in the English Dictionary. If this is not the case, the system administrator has possibility to specify the synonyms.

4.3.1. The Pre-defined data structures

The proposed approach employs a set of predefined training structures. The primary benefit of these training sets is that they can be expanded or appended when the intelligent information system discovers some new knowledge. The significant training sets used are: Conjunction set (C_T) , Expression mapping (E_{map}), Stop-words set (S_W) and Display part set (D_{PT})

The Conjunction training set (C_T) consists of the list of generally used Conjunctive clauses like hailing, having, when, where, who, whose, with etc. These conjunctive clauses determine the exact Query definition. When the system encounters a relatively new conjunctive clause, it is appended to the existing training set.

The trained stop word set (S_W) contains the list of all common stop words that are likely to occur in a user typed query.

Display part training set (D_{PT}) contains the list of words like Detail ,details, display, displays, extract, find, out, get, get me, Gets ,limit, limits , list, lists, produce ,select, show, summarise, summarize, view for determination of Display part

The Expression mapping set (Emap) contains the list of commonly used conditional clauses(Figure 2) and their associated mathematical symbols. It acts as a look up table to locate the SQL defined mathematical operators.

Expression Mapping	
Element	**Semantic set name**
greater than	>
less than	<
greater than or equal	>=
less than or equal	<=
not equal	<>
greaterthan	>
lessthan	<
greater than or equal to	>=
less than or equal to	<=
like	like
equal	=
order by	order by
equal to	=

Figure 2 Expression mapping set

5. EXPERIMENTS AND RESULTS

Here, we demonstrate the capabilities of our system via experiments. The experiment deals with generation of SQL for a natural language query and compare the performance of the our interfaces on the basis of accuracy.

5.1. Experiments

We ran our experiments on NORTHWIND database sample that is shipped with MS SQL server and MS ACCESS. The standard eight tables selected. The figure 3 below shows and overview of tables, fields, and joins in the NORTHWIND database.

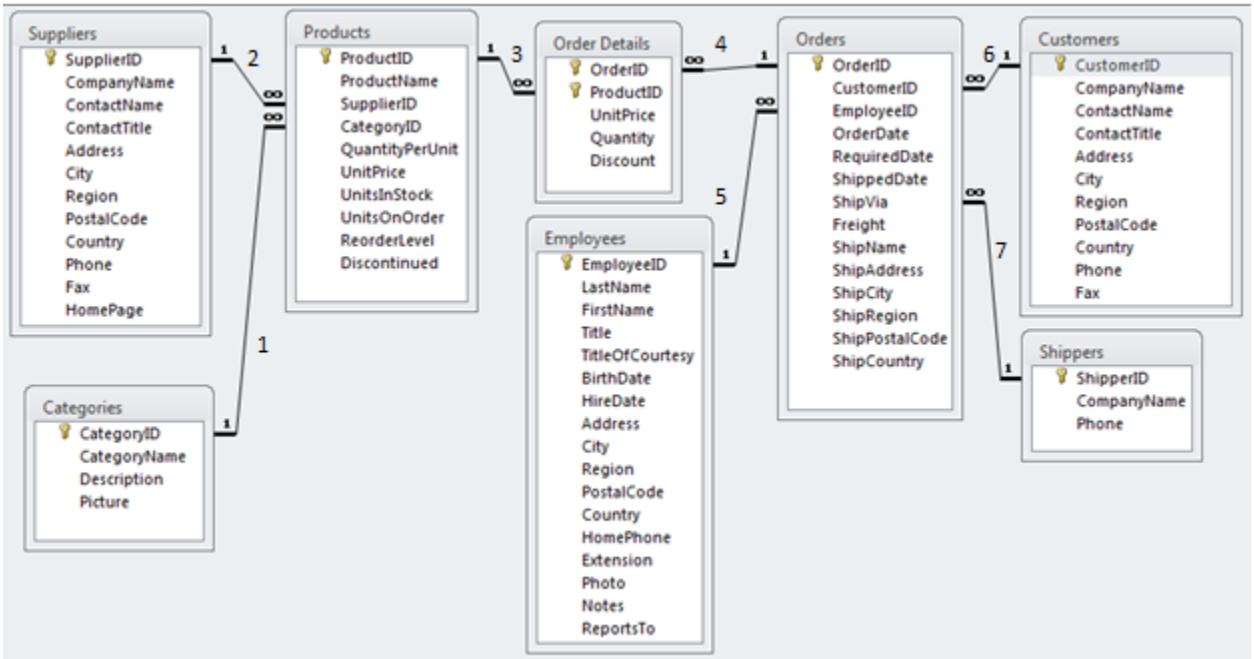

Figure 3 Schema of Northwind Database

Experiment is related to the generation of the SQL statement from the English natural query. In this section , we demonstrate our system capabilities in generating the different form of the SQL statements depending on the structure of the English natural queries. A general query simple is a query where there are no specifics for the attributes list, conditions, relationship, etc… Table-1 depicts some examples.

Table-1: Simple queries.

User query	Generated SQL
Display all Employees	*SELECT * FROM employee*
Get me supplier details	*SELECT * FROM suppliers*
Who are our customers	*SELECT * FROM customers*
What are product categories	*SELECT * FROM categories*

A specific query is a query with some certain attributes. So, the selection of the attributes is done from tables. Table-2 depicts some examples of SQL with the specific attributes.

Table-2: Examples of specific attributes.

User query	Generated SQL
tell me our supplier location	*SELECT address FROM suppliers*
display employee address	*SELECT address FROM employee*
Display names of suppliers	SELECT lname, fname FROM *supplier*
what are our employee name?	SELECT *employee.name* FROM *employee*
Get me the name of categories	SELECT categoryname FROM *categories*
Display employee name and location ?	*SELECT lname,fname, address FROM suppliers*
tell me our employees names and addresses	*SELECT lname,fname, address FROM employee*
display employees addresses and names	*SELECT lname,fname, address FROM employee*

A conditioned query is a query that will select some certain tuples of the database giving some specific criteria. Table-3 depicts some examples of SQL statement with WHERE clause. Sometimes the user would like to inquire about some certain attributes that satisfy some given condition(s).

Table- 4 depicts some examples of SQL statements with specific attributes and conditions.

Table-3: Examples with WHERE clause.

User query	Generated SQL
Get the details of employees who are located in LONDON	*SELECT * FROM employee WHERE city='LONDON'*
Get supplier details for supplier id is S1	*SELECT * FROM supplier WHERE suplierid='S1'*
Get employee details not located in PARIS	*SELECT * FROM employee WHERE city NOT IN 'PARIS'*
List employee details who are located in LONDON or SEATTLE	*None*
Get the details of employees in LONDON	*None*
Find the products which have units in stock 20 and unit price is 18	*SELECT * FROM products WHERE unitsinstock = 20 a nd unitprice = 18*
tell me product details for product name TOFU	*SELECT * FROM product WHERE productname = 'TOFU'*

Table-4: Examples with specific attributes with WHERE clauses

User query	Generated SQL
Get names of employees whose hometown is PARIS	*SELECT lname,fname FROM employee WHERE city ='PARIS'*
Get names of employees in PARIS	*None*
display employee address for employee id is E101	*SELECT address FROM employee WHERE employeeid='E101'*
tell me unit price for product name TOFU	*SELECT unitprice FROM product WHERE productname ='TOFU'*
Get product id of product for qty ordered greater than 30 and unit price is less than 10	*SELECT productid FROM orderdetails WHERE quantity > 30 AND unitprice < 10*

Our system can also deal with queries that need data from more than one relational database. In such case the system has the capabilities to perform the appropriate joins to retrieve the required data. Table- 5 depicts some examples of SQL statements which involves more than one table

Table-5: Examples involving more than one table

User query	Generated SQL
Get supplier details whose product id is P1	SELECT * from Supplier, Products where supplier. Supplierid=products.supplierid AND productid='P1'
display customer id whose order date is 01/10/1996	SELECT *customerid* FROM *customers,orders WHERE coustomers.cusomerid=orders.customerid AND orderdate = '01/10/1996'*
List all suppliers whose supply product category Beverages	*SELECT * FROM products, suppliers ,products WHERE categoryname ='Beverages' AND categories.categoryid = products.categoryid AND suppliers.supplierid = products.supplierid*
List all products supplied to country Germany	*SELECT * FROM products , suppliers WHERE suppliers.city='Germany' AND suppliers.supplierid = products.supplierid* Result Incorrect

5.2. Results

Much experiment on a trial basis has been conducted on our system. The trials have given accurate and satisfactory results when the generated SQL statements have been run against the used database. We have compared our interface with natural language interface which is commercially available.

There are many Natural Language Interfaces available for commercial use and each claim to perform better than the other. We have selected Microsoft English Query[19] to compare with our interface. Experiments are conducted to compare the performance of these two interfaces on the basis of accuracy. We have evaluated the performance of English Query and our system and reach to a conclusion as to which one performs better. The experiment was to test the questions in both the applications using only the basic model. This tested the capabilities of Our Interface and EQ to automatically extract relationships from the underlying database. Figure 4 shows the results of this test. Our interface gives correct results for most of the questions and English Query does not. This is because English Query does not extract all of the relationships and requires refining of the model by adding relationships.

The relationships in the EQ were added for only those queries which failed the first test. The performance of English Query improved by adding relationships. The results clearly illustrate that Our Interface compares favorably with Microsoft English Query.

Interface	No of question asked	No of correct result	No of incorrect results
Our Interface	20	15	5
EQ	20	8	12

Figure 4 Test Results

In our system, the basic model was used and no modifications were made. This shows that our system is effective and automatically extracts relationships from database. Whereas EQ builds up a model with only few basic relationships and so requires a lot of modification and refinement. This is tedious and involves a lot of work.

6. REFERENCES

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

[2] Raymond J. Mooney, Learning Language from Perceptual Context:A Challenge Problem for AI, American Association for Artificial Intelligence (2006).

[3] VILIB Virtual Library (1999), www.islp.uni-koeln.de/aktuell/vilib/

[4] Chae, J., Lee, S.: Frame-based Decomposition Method for Korean Language Query Processing. Computer Processing of Oriental Languages (1998)

[5] Popescu, A.:Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability, University of Washington (2004)

[6] Waltz, D.: An English Language Question Answering System for a Large Relational Database. Communications of the ACM (1978)

[7] Microsoft TechNet., chapter 32- English Query Best Practices (2008), www.microsoft.com/technet/prodtechnol/sql/2000/reskit/part9/c3261.mspx?mfr=true

[8] Popescu, A., Etzioni, O., Kautz, H.: Towards a Theory of Natural Language Interfaces to Databases. In: Proc. IUI-2003, Miami, USA (2003).

[9] ELF Software, ELF Software Documentation Series (2002), www.elfsoft.com/help/accelf/Overview.htm

[10] SQL-HAL, www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura/

[11] Androutsopoulus, I., Ritchie, G., Thanish, P.: MASQUE/SQL, An Efficient and Portables Language Query Interface for Relational Databases, Department of Artificial Intelligence, University of Edinburgh (1993)

[12] Minock, M.: A STEP Towards Realizing Codd's Vision of Rendezvous with the Casual User.In: Proc. 33rd International Conference on Very Large Databases (VLDB-2007), Demonstration Session, Vienna, Austria (2007)

[13] Minock, M.: Natural Language Access to Relational Databases through STEP. Technical report, Department of Computer Science, Umea University (2004)

[14] Bagnasco, C., Bresciani, P., Magnini, B., Strapparava, C.: Natural Language Interpretation for Public Administrations Database Querying in the TAMIC Demonstrator. In: The Proc. Second InternationalWorkshop on Applications of Natural Language to Information Systems (1996)

[15] Chu, W., Yang, H., Chiang, K., Minock, M., Chow, G., Larson, C.: Cobase – A Scalable and Extensible Cooperative Information System. Journal of Intelligent Information System 6,253–259 (1996)

[16] Alshawi, H., Carter, D., Crouch, R., Pulman, S.: CLARE: A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine. Technical report CRC-028(1994)

[17] Binot, J., Debille, L., Sedlock, D., Vandecapelle, B.: Natural Language Interfaces: A New Philosophy, SunExpert, Magazine (1991)

[18] Boldasov, M., Sokolova, G.E.: QGen – Generation Module for the Register Restricted In-BASE System. In: Computational Linguistics and Intelligent Text Processing, 4th International Conference, vol. 2588, pp. 465–476 (2003)

[19] Microsoft English Query Tutorials available with standard installation in SQL SERVER 7.0 or higher